أساسيات معالجة الصور الرقمية

تأليف
الأستاذ المساعد
هند رستم محمد شعبان
Hind_restem@yahoo.com

الفهرست

الفصل الأول
المقدمة

الفصل الثاني
تحليل الصورة

الفصل الثالث
تحسين الصورة الرقمية

الفصل الاول

المقدمة

1-1 معالجة الصورة:

معالجة الصورة(Image Processing) هي تمثيل للصّور الثنائية الأبعاد على الحاسوب بواسطة الصفر و الواحد (01) ، و تتكون كل صورة رقمية على الحاسوب من البكسل(Pixel) وهو أصغر وحدة في الصورة و كل صورة تحتوى على صفوف و أعمدة من البكسلات و كلما زادت عدد البكسلات كلما كانت الصورة أوضح.

تعرف معالجة الصورة أيضا بأنها أحد فروع علم الحاسوب (المعلوماتية) ، تهتم بإجراء عمليات على الصور بهدف تحسنيها طبقاً لمعايير محددة أو استخلاص بعض المعلومات منها.

يتألف نظام معالجة الصور التقليدي من ستة مراحل متتالية وهي على الترتيب:

1. استحصال الصورة (image acquisition) بواسطة حساس ضوئي(على سبيل المثال آلة تصوير،حساس ليز وغير ذلك)
2. المعالجة الابتدائية (pre-processing) كتصفية الصورة من التشويش
3. تقطيع الصورة (segmentation) لفصل المعلومات المهمة على سبيل المثال أي جسم في الصورة عن الخلفية
4. استخلاص المميزات (features extraction) أو الصفات
5. تصنيف المميزات (classification) و ربطها بالنمط الذي تعود أليه و التعرف على الأنماط
6. فهم الصورة (image understanding)

وتستخدم نظم معالجة الصورة في الكثير من التطبيقات ولاسيما تطبيقات التحكم الآلي ،الإنسان الآلي والخ.

يمكن أن تستخدم معالجة الصورة الرقمية لتحديد التفاصيل الخاصة بإضاءة صورة معينة أو تحديد هيكل الصورة بالمقابل من خلال قيم (مايكرو سكوب) دقيقة والتي من خلالها يتم استخراج قيم للصورة الرقمية يمكن أن تأخذ من خلال خرائط للصور ، مثلا صور الستلايت أو صور القمر الصناعي .

كما يمكن أن تحول وتعالج الصور من خلال الحاسوب بصيغ رقمية من خلال عدة مصادر :

Camera digital -1

الكاميرا الرقمية : تحول الصور إلى صور رقمية ذات دقة متناهية وشدة الوضوح اللوني للصور.

Any photo Graph -2

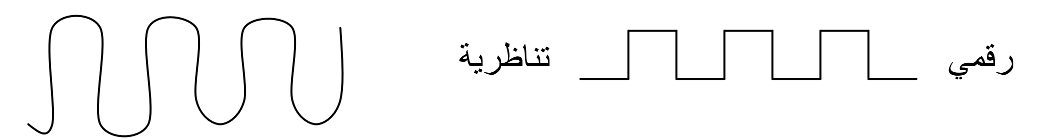
أي فوتوكراف يمكن أن يحول الصورة إلى صيغة رقمية مثل ال Scanner

Photoshop -3

برنامج يضيف بعض التعديلات على الصورة سواء إضافة أو حذف أو أي تعديل يناسب ذلك .

4- برنامج الكورول (Corel Draw):

Note : البيانات المدخلة هي تناظرية وليست رقمية والشكل ادناه يوضح ذلك

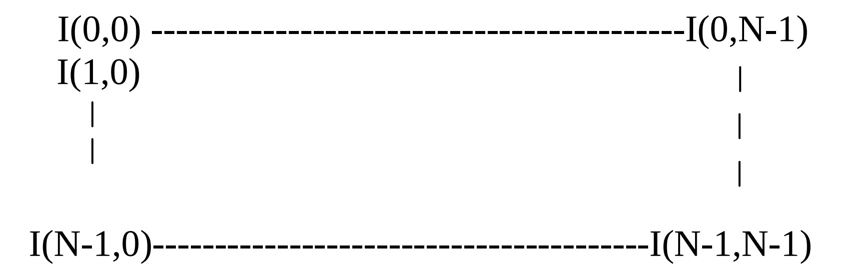


شكل(1):الإشارة الرقمية والتناظرية

أنواع أو حقول معالجة الصورة

1- Image restoration **أعادة مكونات الصورة**
2- Image enhancement **تحسين الصورة**
3- Image compression **ضغط الصورة**
4- Image segmentation **تقطيع الصورة**

الصورة الرقمية سوف تأخذ الصيغة المحددة بمصفوفة (I(x,y أي تكون مصفوفة الصورة هي (I(r,c حيث تمثل بقيمتين هي point أو pixel
(r,c) هي لنقطة واحده
(I(r,c هي لنقاط المصفوفة كاملة لكل الأعمدة والصفوف هذه النقاط تمثل بمصفوفة تأخذ أبعاد (tow dimension) أي ذات بعدين



حيث أن الصورة هنا سوف تحول إلى صيغة رقمية بطريقتين

1- الطريقة الأولى : الإحداثيات الحيزية Spatial coordinate
حيث أن الصورة تكون على شكل عينات (Image sampling) تعتمد على قيم الصورة الخاصة (أي مباشرة قيم الصورة نفسها)

2- الطريقة الثانية Amplitude domain

تعتمد هذه الطريقة على كميات المستوى الرمادي يعني (Gray level quantization) أي تعتمد على المستويات : مثلا 3 مستويات فيكون عدد المستويات الرمادي يمكن إيجاده بالقانون التالي وهو

عدد المستويات الرمادية

number gray levels(Ng)= 2^n……………….(1)

$2^3 = 8$

لهذا السبب سوف تتولد لدينا مصفوفة ذات حجم I(N*N) حجمها N*N وهي أبعاد الصورة .

يستخدم القانون أدناه لإيجاد عدد البتات التي تحتاجها مصفوفة الصورة الرقمية .

B = N * N * M ………………………………….(2)

حيث أن (M) هي عبارة عن عدد البتات المطلوبة والتي تمثل كل مستوى رمادي .
عدد البتات الكلية للصورة هو **B**
وعدد بتأت كل مستوى رمادي هو **M**
N,N هي أبعاد الصورة

مثال // إذا كان لديك 6 بت يعني (128 *128) مصفوفة صورة

1- أوجد عدد المستويات الرمادية المطلوبة للصورة
2- أوجد عدد البتات للصورة كاملة

الحل//

1- لاستخراج عدد المستويات الرمادية **$Ng = 2^n = 2^6 = 64$**

-2 **B = N * N * M**

B= 128 * 128 * 6

B= 98304 8 bit : عدد البتات في المستوى الواحد معطى بالسؤال

يجب أن تكون هنالك دقة في الخزن وعدد البتات للخزن هي 6 بتأت أي 6 مراتب وبجانب أخر الرقم المستخرج كبير جدا لذلك يجب أن يحول إلى
كالأتي $9.8304 * 10^4$
10^4 لأن زحفنا الرقم أربع مراتب لكي أحصل على أقل رقم للخزن .

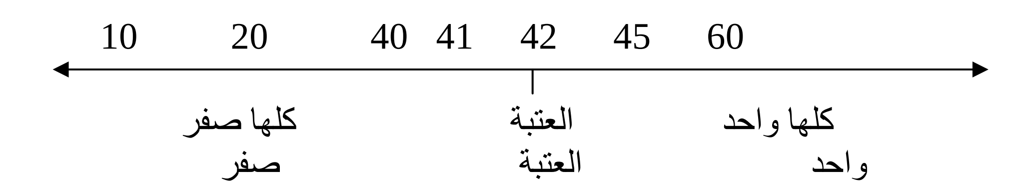
2-1 : أنواع الصور (Type of Images):

تقسم الصور الرقمية إلى الأنواع التالية

1- الصورة الثنائية (Binary Image)

قيم الصورة بعد تحويلها تكون كلها أما أصفار أو واحد أما أسود أو أبيض يمكن تحويل كل أنواع الصور إلى الصور الثنائية عن طريق ما يسمى بالعتبة (Threshold) قبل العتبة هو صفر وبعد العتبة هو واحد

مثال//



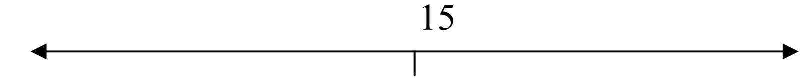
نأخذ العتبة هنا على طريقتين

12	14	28
40	5	9
15	20	7

الطريقة الأولى : هي أن نختار أي رقم من الأرقام وهو مثلا الرقم 15 فكل رقم أقل من ال15 هو صفر يمثل بالمصفوفة وكل رقم أكبر من ال15 هو واحد

فتصبح تمثيل المصفوفة كالأتي

0	0	1
1	0	0
1	1	0



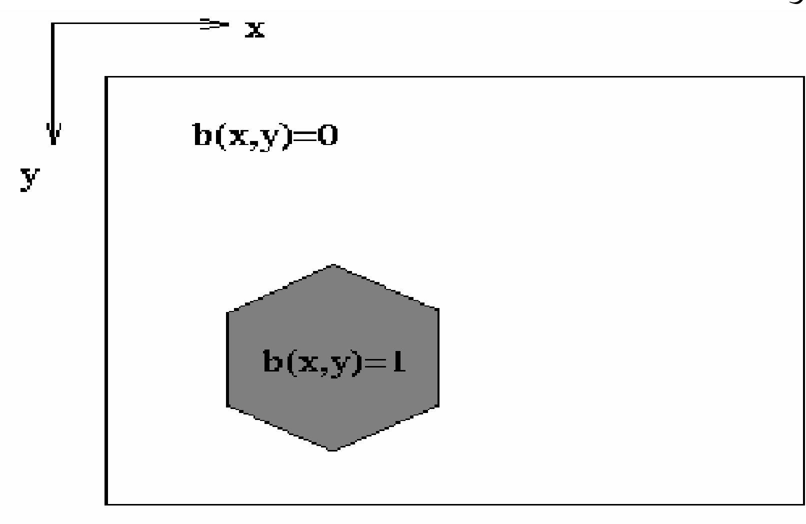
الطريقة الثانية : نأخذ أقل رقم من المصفوفة وأكبر رقم من المصفوفة ونجمعهم ونقسمهم على 2 وبالمثال أقل قيمة هي 5 وأكبر قيمة هي 40 إذن

$40 + 5 = 45 \div 2 = 22.5$ لذلك تكون العتبة 22 وتكتب المصفوفة

العتبة هنا 22

0	0	1
1	0	0
0	0	0

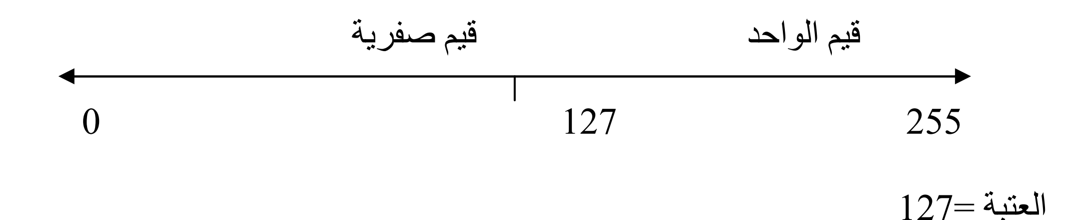
الصورة الثنائية هي أبسط أنواع الصور تتمثل باللونين الأبيض والأسود أو يرمز له بالصفر أو الواحد ، فالصورة الثنائية يمكن أن يشار أليها بالمعنى (1 bit per pixel) كل بكسل يأخذ قيمة واحده مثل الصفر بكسل واحد ، وكذلك يمكن تتكون هذه الصورة من أنواع الصور الأخرى مثلا صورة المستوى الرمادي وذلك باستخدام ما يسمى بالعتبة
كل باند (Band) يقابل لون واحد ، ملاحظة الصورة الرقمية الثنائية لا تمتلك لون ثالث بل فقط الأبيض والأسود



شكل(2):صورة ثنائية

2- النوع الثاني Gray _Scale _Image

يمثل هذا من الصور على أساس لون واحد للصورة أو ما يسمى بال (Monochrome) حيث يمتلك معلومات عن الإضاءة للصورة فقط ولا يمتلك معلومات عن اللون هذه المعلومات الخاصة بإضاءة مستويات الصورة تمثل ب8bit على الكسل للبيانات لمختلف المستويات يعني 2^8 = 256 يعني من 0 إلى 255 من مستويات الإضاءة المختلفة هذا النوع من أنواع الصور سهل التحويل إلى الصورة الثنائية



العتبة =127



شكل(3): صورة نوع Gray _Scale _Image

3- النوع الثالث Color Image

الصورة الملونة تمتلك موديل معين يتكون من (3 Bands) (كل باند لون واحد) يعني ثلاث ألوان أحادية للصورة كل لون يشار له بتمثيل معين هي الأحمر Red والأخضر Green والأزرق Blue يسمى (RGB) كل لون يأخذ 8 بت إذن الصورة الملونة تمتلك 24 بت (24 bit per pixel)

شكل(4):صور ملونة

4- النوع الرابع Multi spatial Image

الصورة متعددة الأطياف : تأخذ من قبل الآلات تصوير خاصة توجد فيها العديد من الBand الباندات قد يصل في بعض الأحيان إلى مئات من الباندات وعند العمل عليها تسقط هذه الباندات بطريقة معينة تسمى (Mapping) التخطيط أو التبويب بحيث تقابل النوع الثالث من الصور .

يعني تصبح من متعددة إلى صورة ملونه (أي يجمع مثلا تدرجات الأحمر تدمج باللون الأحمر) وذلك بعملية الإسقاط على الصورة لكي تصبح للصورة ثلاث ألوان أساسية فقط .

3-1 اهمية معالجة الصورة الرقمية

توجد أهمية كبيرة للمعالجة الرقمية للصور في ميدان(**إدراك الصورة**) أي عندما نحاول مثلا أن نجعل الحاسوب أو الرجل الآلي يفهم الصورة ولها أهمية في ميدان(**التعرف على الأنماط**) أو الأشكال.

أن للتعرف على الأنماط أهمية كبيرة في المعالجة الآلية للصور التي تلتقطها المكوكات لسطح الأرض حيث يمكن استخدامه في المجالات العسكرية وفي الملاحة إعتمادة على خرائط أو صور من الأرض.

تعامل الصورة كإشارة ويتم تطبيق طرائق المعالجة الرقمية للإشارة عليها من خلال المرشحات (الفلاتر) التعرف على أنماط أو أجسام ضمن الصورة مثلا تحسس وجود أورام في صورة شعاعيه .

عندما تلتقط الصور، تقوم الكاميرا بحفظها على الوسيط الرقمي ضمن أحد النساقات التالية:

JPEG -1

النساق JPEG اختصار للعبارة ,(Joint Photographic Experts Group) ويعتبر النساق الأكثر شعبية وانتشاراً لا سيما لعرض الصور على الانترنت. المصطلح "JPEG" يستخدم عادة لوصف النساق الملفي JFIF والذي هو اختصار ل .(JPEG File Interchange Format) إن JFIFهو الشكل الفعلي للملفات الحاوية على صور مضغوطة وفق نظام .JPEG في الوقت الحالي تستخدم ملفات JFIF الحديثة نفس التمدد .jpg ولكن هناك اتجاه بتغيير التمدد إلى .Jif في الأنظمة المستقبلية.

يستعمل JPEG آلية ضغط متغيّرة، حيث تستطيع التحكم بدرجة الضغط عند التخزين، للحصول على حجم ملف مناسب، حتى أنه يمكنك الحصول على حجم ملف صغير جداً ولكن طبعاً مع ضعف في جودة الصورة .

يدعم النساق JPEG نظام عمق لوني لغاية (24 بت 16) مليون لون، في حين أن العمق اللوني للنساق Gif محصور ب 8 بت 256 لون.

يتم الضغط عبر وحدات (بلوكات) تتألف من ثماني بكسلات. تستطيع رؤية هذه البلوكات عندما تختار أعلى درجة من درجات الضغط، أو عندما تقوم بتكبير الصورة إلى قياس كبير جداً.

يعمل JPEG وفق إلية ضغط ثنائية المراحل .هذا يعني أنه يحتاج إلى وقت أطول من أجل تحميل وعرض الصورة. بعد عدد من المرات، تضيع التفاصيل الدقيقة والتدرجات اللونية.

يفضل حفظ الصور الأصلية وفق نساقات غير مضغوطة مثل TIFF أو BMP وبأقصى عمق لوني متوفر. عندما تقوم بحفظ الصورة وفق النساق ,Jpeg فإن التغيير الحاصل على الصورة لا ينعكس على الشاشة مباشرة، ولكن فقط بعد أن تقوم بتحميل الصورة من جديد.

TIFF - 2

TIFF اختصار ل (Format Tag Image File)، صممتها شركة آلدوس Aldus في الأصل لحفظ الصور المستوردة من الماسح الضوئي (Scanner) أو من برامج المعالجة. أنتشر هذا النساق بشكل واسع، وشاع كنساق نقل الصور دون أن يكون مرتبط بماسح ضوئي معين أو طابعة أو برنامج معالجة. النساق TIFF يحظى بشهرة واسعة أيضاً مع تطبيقات النشر الاحترافية. هنالك عدة صيغ للنساق TIFF تدعى توسعات (extensions) ، من هنا تظهر بعض المشاكل عند محاولة تحميل أحدها عن طريق الآخر. بعض التوسعات تتعامل بآلية ضغط من النوع LZW التي لا تضعف الصورة بتاتاً. نساق TIFF يدعم عمق لوني 24 بت كحد أقصى.

CCD RAW -3

تقوم الكاميرا بمعالجة بيانات الصورة التي تسجلها الخلية الضوئية CCD وحفظها في أحد النساقات. بعض الكاميرات تسمح بحفظ البيانات الخام (غير معالجة وغير مضغوطة) في نساق يسمى CCD RAW أو اختصارا .(CRW.) هذه البيانات تحتوي على كل شيء التقطته الكاميرا. وبدلاً من معالجة هذه البيانات داخل الكاميرا، حيث قوة المعالجة وحيّز العمل محدودين. تتم معالجة البيانات الخام وتحويلها إلى الصورة النهائية عن طريق حاسوب خارجي. أن حيّز العمل الواسع وقوة المعالجة التي يتمتع بها الحاسوب الخارجي من شأنها التأثير إيجاباً على جودة الصورة في المحصلة النهائية

أحد أهم خصائص ملفات النساق CCD RAW الناتجة عن كاميرا رقمية- صغر حجم الفايل وبنسبة تصل إلى 60% أقل من حجم الفايلات من النساق RGB TIFF غير المضغوطة (في حال كانت كثافة التسجيل Resolution لكلا النساقين متساوية). صغر حجم الملف (مع الحفاظ على جودة الصور) يتيح للكاميرات الرقمية اختصار الزمن بين اللقطات.

إن النساقCRW يسجل بيانات الخلية الضوئية وبواقع بايت لكل بكسل ويسجل بيانات توازن اللون الأبيض , White Balance وخريطة التباين Contrast mapping وغيرها من البيانات الضرورية ,التي تساعد في الحفاظ على دقة الألوان وغيرها من أمور مهمة عند معالجة الصورة.

أن بعض الكاميرات الرقمية الحديثة تسجل الصور في نساق CCD RAW بعمق لوني 10بت/قناة ولأربع قنوات . (C-M-Y-G) بينما تعمل تطبيقات المعالجة على تحويلها إلى نظام RGB بعمق لوني كلي 24 بت. من المتوقع أن تتحول الكاميرات المستقبلية إلى نظام تسجيل بمستوى 12-بت لكل قناة، الأمر الذي سيؤدي إلى تحسين التدرجات اللونية للصورة.

4-1 آلات التصوير الرقمية Digital Cameras

يتم التقاط الصور بالكاميرا الرقمية بنفس الطريقة التي تلتقط بها الصور بالكاميرا العادية الفرق هو آن الكاميرا الرقمية لا تستعمل الفيلم العادي وبدلا منه فان الصور بتم تسجيلها إلكترونيا وتخزنها في الذاكرة الداخلية للكاميرا إذا كانت تحتوي على ذاكره داخليه أو أن تخزن على بطاقة ذاكره خارجي (وهو في هذه الحالة يمكن تشبيهه بفيلم إلكتروني) أو أن يتم تسجيله على قرص لين كقرص الحاسوب العادي Floppy Disk عاده فانك تستطيع رؤية الصور بكامل ألوانها على شاشة الكريستال السائل LCD الداخلية للكاميرا تستطيع توفير جزء من ذاكره الكاميرا لتلتقط عليها صورة أخرى.

ويجب أن نأخذ بنظر الاعتبار بعض الأمور المهمة:

أ- جوده الصورة

تعتمد جوده الصورة جزئيا على كميه التفاصيل (Resolution)والتي تستطيع الكاميرا إيجادها ويمكن قياس ذلك بعدد الحبيبات Pixels وهي تلك القطع المتناهية الصغر والحساسة للضوء في الكاميرا . إن قدره الكاميرا على إعطاء تفاصيل حادة يمكن معرفتها مسبقا إما بمجموع عدد تلك الحبيبات أو القطع الصغيرة(Pixels) الموجودة بالكاميرا مثلا307200 أو بقياسات الخطوط الأفقية والرأسية 480 × 640 (وهو يساوي نفس القياس الأول 307200) وعلى العموم فكلما تواجدت بكسلات أكثر بالكاميرا كانت الكاميرا أفضل.

يمكن للكاميرات أن تستخدم تقنية ضغط المعلومات (Data Compression) لتخزين الصور لتوفير استخدام جميع القطع أو الحبيبات الصغيرة المسماة بكسل Pixels وفي هذه الحالة تقل جوده الصورة ولكن هذا يعني استخدام مساحة ذاكره اقل . إن جوده الصورة تعتمد على درجه جوده العدسة وعدد الألوان التي تستطيع الكاميرا أن تستجيب لها.

ب- ذاكرة الكاميرا

يتم تخزين الصور في كاميرات الديجيتال (الرقمية) في ذاكره داخليه أو في ذاكره خارجية . بعض آلات التصوير الرقمية تستخدم الطريقتين وتقاس الذاكرة بوحدة قياس تسمى ميجابايت Megabyte . إن حجم الذاكرة أو كميه الميجابايت التي تأخذها كل صوره تختلف حسب اعتبارات عديدة تعتمد على حده التفاصيل في الصورة أو ما نطلق عليه هنا بقوه التحديد Resolutionوكذلك على عدد الألوان .

أن كميه الذاكرة التي تستهلكها كل صوره تختلف باعتبارات عديدة تعتمد على درجه تفاصيل الصورة أو قوه التحديد Resolution وكذلك على استخدام تقنية ضغط المعلومات Data Compressionوعدد الألوان . نتيجة لهذا فانه لا يعني بالضرورة أن الذاكرة الكبيرة تستطيع تخزين عدد اكبر من الصور . فبحسب قوه التحديد أي حده التفاصيل للصورة وجودتها يمكن أن تخزن الكاميرا صورا قد تكون عشر صور وقد تصل إلى تسعين صوره في الذاكرة التي تأتي معها .

ج- الذاكرة الخارجية للكاميرا

يتوفر نوعين من بطاقات الذاكرة الخارجية وذلك بحسب طاقه تخزينها . الأولى طاقه التخزين أربع ميجابايت والثانية ثماني ميجابايت . يتم التقاط تسجيل الصور عليها فإذا امتلأت فان باستطاعتك تفريغها إلى الحاسوب .

بعض آلات التصوير تستخدم القرص اللين العادي 3.5 Inch Floppy Disk ولكن قوه تخزينه قليل 1.4 ميجابايت بالإضافة إلى أن حجم الكاميرات في هذه الحالة يكون اكبر ولكن قدره التخزين القليلة هذه يعوضها ثمن القرص اللين بينما بطاقة التخزين من النوع الأول قد يصل إلى خمسين باوندا.

د- تفريغ الصور

يتم تفريغ الصور إلى الحاسوب بمساعده برنامج خاص بذلك وهذا البرنامج يأتي عاده مع الكاميرا عند شرائها. وعمليه التفريغ سهله. ومهما كان نوع أو مصدر البرنامج المستخدم فان عمليه التفريغ تتم باستخدام سلك متصل بالحاسوب من الخلف من النوع التسلسلي Serial Port ويأتي السلك عاده مع الكاميرا عند شرائها وتحتاج العملية لعده دقائق لنقل كل الذاكرة.

يمكن أن تتم عمليه النقل من الذاكرة الخارجية بواسطة جهاز غالي الثمن نسبيا وهذه طريقه أسرع.

5-1 تخزين الصور الرقمية في الحاسوب

يحتاج تخزين الصور إلى مساحة كبيره على قرص الذاكرة الصلب Hard Disk الموجود بداخل الحاسوب وأن يكون الحاسوب بحد أدنى من المواصفات أهمها أن يكون مزودا بذاكره "رام RAM "لا تقل عن 16 ميجابايت .

أن رؤية الصور بنوعيه جيده تتم من خلال يكون تزويد الحاسوب بذاكره فيديو خاصة Video RAM (V- RAM)لا تقل عن 2 ميجابايت . حيث إن سرعة المعالج المركزي Central Processing Unit "CPU"العالية مهم واذا أردنا معالجه الصور كإجراء بعض التغييرات عليها فإنها ضرورية ، وكلما كان المعالج أسرع كلما كانت معالجه الصور أسرع.

هناك عده طرق لمشاهده الصور المأخوذة بواسطة الكاميرا الرقمية ، منها أن يتم وصل الكاميرا بجهاز التلفزيون ورؤيتها على الشاشة كما يمكن رؤيتها على شاشة الحاسوب ويمكن وضعها في إي موقع على الإنترنت.

يمكن طباعه الصور فبعض الكاميرات تستطيع أن توصل بالطابعة مباشرة وهناك أنواع أخرى يلزم أن يكون ذلك عن طريق الحاسوب . يمكن أن تكون الطابعة من أي نوع ولكن الطابعات الرقمية الخاصة بهذا الغرض تعطي صورا أفضل بكثير.

إن جوده الصورة المطبوعة تعتمد على نوع الطابعة ونوع الورق المستعمل وهناك أنواع من الورق الفوتوغرافي لهذا الغرض يعطي أفضل النتائج ولكنه غالي الثمن..أن تكلفه الصورة في الحالة الأخيرة قد تكون أكثر من ضعف تكلفه طباعه الفيلم المعتاد حتى مع تكاليف تحميضه. إن الورق العادي بالطبع ارخص بكثير ولكنه سهل التلف وجوده الصورة لا تكون جيده ولا تبدو الصورة طبيعية كالصور العادية التي اعتدنا أن نراها.

إذا تفحصنا الصور المطبوعة من كاميرا الديجيتال عن قرب فستجد أنها متكونة من نقاط من الألوان وينطبق هذا على الصور المطبوعة بأي نوع من الطابعات حتى وان كانت طابعه رقميه متخصصة ، وكلما كان عدد النقاط أكثر وذات حجم اقل أي Resolution أعلى كلما كانت الصور أفضل وطبيعي أن نوع الكاميرا المنتجة للصورة لها تأثيرها في هذا المجال فالنوعيات ذات قوه التحديد الضعيفة يمكن أن تكون صورها مقبولة لعمل النشرات المطبوعة .

كثيرا ما يفشل التحكم التلقائي بالتعريض في حساب كميه الإضاءة الصحيحة اللازمة في حاله وجود خلفيه للصورة ذات إضاءة قويه ولذا فان توفر إمكانية زيادة التعريض في الكاميرا ،و تقدر طاقه ذاكره الكاميرا بعدد اللقطات التي يمكن أخذها باستخدام اكبر قدر من الذاكرة المتوفرة. وتوفر إمكانية استخدام كميه من الذاكرة اقل في حاله استخدام قوه تحديد اقل Resolution وكذلك عند عمليه ضغط المعلومات للصورة Data Compression

6-1 مكونات الكاميرا الرقمية

*العدسةLens.

*مصباحFlash.

* ذراع الزوم، وهو ذراع للتحكم بتكبير وتصغير الصورة الملتقطة.

* قاعدة تركيب الحامل الثلاثي (الساند – الاستاند Tripod).

* زر حاجب العدسة، وهو زر يكون استخدامه بالضغط عليه نحو الأسفل يتم من خلاله تسجيل الصورة أو الصوت في الكاميرات المزودة بالميكروفون (Microphone) وبالعادة تكون هذه الكاميرات غير احترافية، وهي مزودة باختيار نحو ضبط مدة التسجيل للصورة والتي تصل إلى عشرة ثواني في بعض الأحيان.

* لاقطة صوتية Microphone (في الكاميرات التي تلتقط الصوت).

* حلقة التركيز البؤري، وهي التي تحدد المسافة مابين العدسة والموضوع المراد تصويره لتحقيق صوره واضحة، وفي اغلب الأحيان تكون الكاميرات الرقمية مزودة بمجسات أو متحسسات تتحسس المسافة وتضبط التبؤر تلقائيا

* مفتاح اختيار وضع التركيز البؤري التلقائي أو التركيز البؤري اليدوي (Focus Auto / manual).

* نافذة خلية كهر وضوئية للفلاش، وهي نافذة تسمح بمرور الضوء على الخلية الكهروضوئية لمعرفة كمية الضوء ومن ثم إرسال إيعاز إلى مبرمج الفلاش لإشعال ضوئه وفق الكمية التي يحتاجها الموضوع المراد تصويره

* مقبس لتوصيل سلك الفلاش الخارجي.

* مقبس لتوصيل سلك الصوت أو الصورة من والى الحاسوب الإلكترونية.

* مقبس لتوصيل التيار الكهربائي المباشر DC in حيث أن أكثر الكاميرات الرقمية بالإضافة إلى أنها تعمل على طاقة البطاريات التي تشغل الكاميرا تكون مزودة بمحولة كهربائية صغيرة تعمل على تزويد الكاميرا بالطاقة الكهربائية المباشرة.

* نافذة خلية كهر وضوئية لشاشة عرض الكريستال السائل(LCD) في الكاميرات التي تحتوي على (LCD)، حيث تتحسس هذه الخلية كمية الضوء المسلط على الشاشة ومن ثم تحدد كمية سطوع الشاشة لكي تكون واضحة، وتكون هذه الشاشة أكثر سطوعا عند تعرضها لضوء الشمس وبشكل تلقائي استنادا إلى هذه الخلية التي توعز إلى كم السطوع.

* أزرار للتحكم بمستوى الصوت (Volume+/-).

* مفتاح اختيار الأوضاع (Movie/Play/Still) وهو زر يحدد اختيار عرض أو تسجيل أو تحرير الصور الثابتة أو المسامع الصوتية أو الصور المتحركة وهي (Play) لعرض أو تحرير الصور (Still) لتسجيل الصور الثابتة والملاحظات الصوتية (Movie) لتسجيل الصور المتحركة.

* زر التركيز البؤري (Focus) فبالإضافة إلى وجود حلقة للتركيز هناك زر في الكاميرا بمجرد الضغط عليه تتم عملية التركيز البؤري تلقائيا.

* مفتـــاح الـــتحكم بالإضــاءة الخلفيـــة لـــشاشة العـــرض الكريـــستال الـــسائل (LCD Back light) حيث يعمل هذا المفتاح على ضبط مستوى سطوع شاشة عرض الكريستال السائل (LCD Bright) فيمكن من خلال رفع أو ضغط الزر تغير السطوع بهذه الشاشة وذلك حفاظا على عين المصور من خلال إعطاء سطوع يوائم ما يرغب المصور بمشاهدته في هذه الشاشة الخلفية أو الجانبية.

* زر التعريض الضوئي التلقائي المبرمج بمؤثرات خاصة (Program Automatic Exposure) حيث يقوم هذا الزر بضبط التعريض للصورة المراد التقاطها وذلك من خلال مبرمج داخل الكاميرا يسمى (Program Automatic Exposure)

* فتحة لإدخال قرص التسجيل، الكاميرات الرقمية لا تعتمد الأفلام الفوتوغرافية المعروفة مثل أفلام (36) أو أفلام (24) أو (120) أو أفلام أخرى عديدة معروفة من قبل المصورين المحترفين بتسميات عديدة، بل أن الكاميرات الرقمية تعتمد أقراص خاصة بتسجيل الصور الرقمية وهي تكون بهيئات عديدة كأن تكون على شكل قرص (Floppy Disk) أو تكون على شكل أخر يشبه الـ (Flash Ram) أو يكون على شكل شريط كاسيت صغير (DV) أو أشكال أخرى كان تكون بطاقة ذاكرة (stick Memory) إلخ.

* ذراع إخراج القرص (Disk Eject)، وهو ذراع يقوم بازلاق مفتاح تأمين إخراج القرص (Eject).
* مفتاح الطاقة (Power)، وهو مفتاح يقوم بتزويد الكاميرا بالطاقة الكهربائية لتشغيل الكاميرا وعملياتها الرقمية، وهذا المفتاح يكون بالعادة مستخدم للتيار الكهربائي القادم من البطارية أو من التيار الكهربائي المباشر.

* زر التحكم، وهو زر أشبه بالدائري في اغلب كاميرات الـ (Digital) حيث يعمل هذا الزر على اختيار الأزرار والصور والقوائم المعروضة على شاشة عرض الكريستال السائل في الكاميرا ويقوم أيضا بتعديل التهيئات، هذا الزر بأربع اتجاهات وعليه إشارات سهم أي أن المستخدم سوف يعرف استخدام هذا الزر من خلال الصورة التي تظهر في الشاشة والاتجاه الذي هو فيه لاختيار العمليات والقوائم المعروضة والأزرار التي تندرج من هذا الزر، فبمجرد الضغط على الزر من الاتجاه الذي تكون فيه القوائم أو الأزرار في الشاشة تظهر مجموعة من الاختيارات لقوائم أو عمليات يرمو لها المصور أو المستخدم.

* زر العرض (Display)، وهو زر يستعرض العديد من المؤشرات المهمة اثناء التسجيل أو التصوير بالكاميرا وهذه المؤشرات إنما هي العمليات التي يرغب المصور دائما بمعرفتها في التصوير للاطمئنان على عمله والتأكد من نجاح التصوير،

والمؤشرات هذه عادة تكون كما يأتي:

1- مؤشر تأمين وضع التعريض الضوئي AE التي تعني التعريض التلقائي أو التعريض الأوتوماتيكي (Automatic Exposure).
2- مؤشر تامين التركيز.
3- مؤشر حدة الصورة.
4- مؤشر وضع التركيز البؤري / مؤشر وظيفة التصوير عن قرب.
5- مؤشر الشحنة المتبقية من البطارية.
6- مؤشر مستوى الفلاش / مؤشر وضع الفلاش.
7- مؤشر وظيفة التعريض الضوئي التلقائي المبرمج بمؤشرات خاصة Program AE / مؤشر الزوم.
8- مؤشر موازنة البياض (balance White).
9- مؤشر مؤثرات الصورة.
10- مؤشر مستوى التعريض الضوئي EV.
11- عمود القائمة ومؤشر إرشاد القائمة وهي تظهر بالعادة بضغط زر التحكم وتختفي بضغط زر التحكم المعاكس.
12- مؤشر وضع التسجيل.
13- مؤشر حجم الصورة.
14- عدد الصور المسجلة.
15- مؤشر السعة المتبقية من قرص التسجيل.
16- 15 مؤشر مدة التسجيل.
17- مؤشر وظيفة عرض التشخيص الذاتي/ مؤشر زمن التسجيل.
18- مؤشر المؤقت الذاتي.
19- مؤشر التعريض الضوئي المتري.

الفصل الثاني

تحليل الصورة الرقمية

1-2 تحليل الصورة (Image analysis):

هي عملية معالجة نقل بيانات الصورة بحيث تستخدم المعلومات الضرورية فقط التي تساعدنا لحل مسالة معينة خاصة بالصورة الرقمية داخل الحاسوب
يمكن أن نحلل بطريقتين هما

1- الرؤية بالحاسوب Computer vision
2- معالجة الصور Image Processing

1- الرؤية بالحاسوب : عند استخدامنا الرؤية بالحاسوب فأننا سوف نحصل على أنتاج نوعي مستخلص لمستويات المعلومات العالية الخاصة للصورة بالحاسوب وهذه المستويات بالمعلومات العالية تتمثل في معاني هي : مثلا اللون والخصائص الأساسية والحدود (الهيكل)

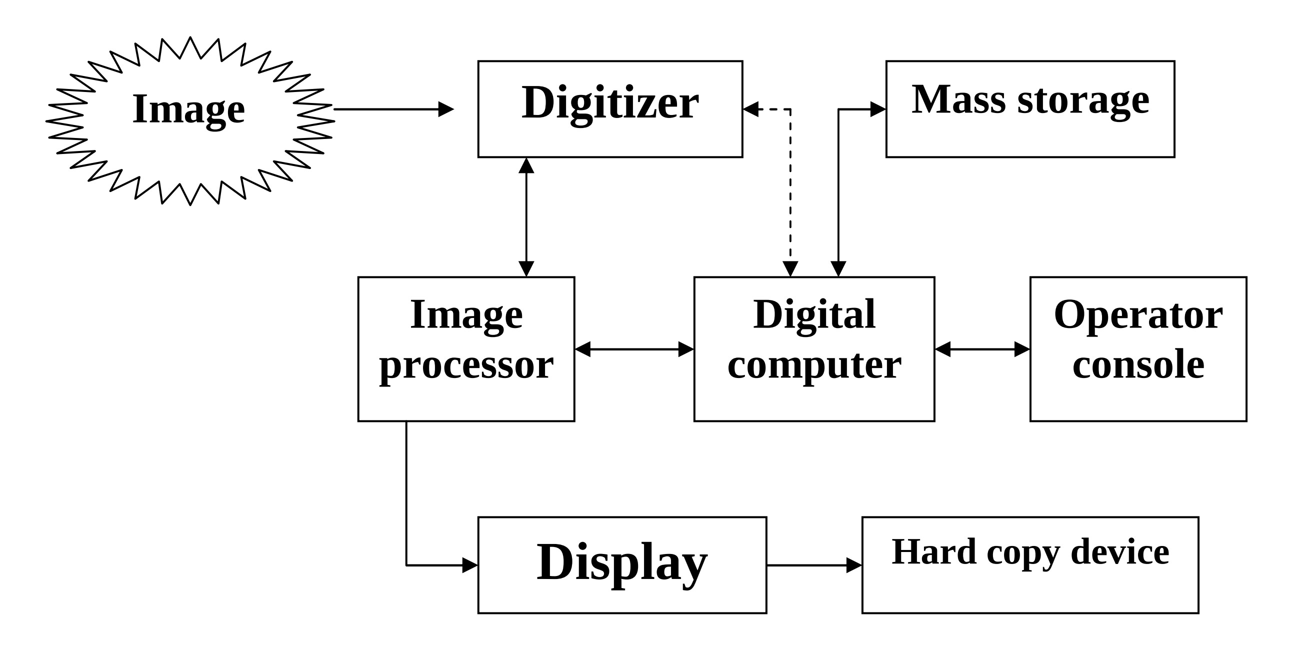
2- معالجة الصور: نحلل الصورة الخاصة بتطبيقات معالجة الصورة باستخدام طرق معينة تحدد المعالجة المطلوبة والمعالم التخصصية . ويوجد نظام أو موديل لتحليل الصورة يتكون من ثلاثة مراحل هي

1- المعالجة الابتدائية Preprocessing
2- تقليل البيانات Data Reduction
3- تحليل الخواص Feature analysis

2-2 عناصر نظام معالجة الصورة الرقمية

تتكون من :

1- معالج الصورة Image Processor
يعد بمثابة القلب لأي نظام معالجة صور ويتألف معالج الصورة الرقمية من الوحدات التالية :
1-الحاسب الرقمي للصورة(تحصيل الصورة) Image Digital computer
2- التخزين Storage
3- المعالجة الصورة Image Processing
4- العرض Display



شكل(5): عناصر نظام معالجة الصورة الرقمية

2- المرقمات Digitizer

مرقم الصورة (يحول الصورة إلى رقمية) هو عبارة عن وحده تقوم بتحويل الصورة إلى تمثيل رقمي بالحاسوب الالكترونية وتوجد العديد من أجهزة الإدخال الأكثر شيوعا المستخدمة مع المرقمات مثلا

1- مقياس الكثافة الضوئية الدقيق (يعتمد على الكثافة الضوئية) .
2- ماسحات النقطة الطائرة (ما هو المركز للضوء) .
3- محللات الصورة (تحليل الصورة كاملة) .
4- كاميرات الفيديو كون (كاميرا + فيديو مثل الكاميرا الرقمية).
5- مصفوفات أنصاف النواقل الحساسة للضوء .

أي نقسم الشاشة إلى أنصاف أقطار ونختار أي نصف قطر نريد .
النقطة الأولى والثانية يتطلبان أن تكون الصورة التي يراد ترقيمها شفافة مثل النسخة السالبة للفلم Film أو النسخة المطبوعة .

أما محللات الصورة أو كاميرات الفيديو كون أو مصفوفة أنصاف النواقل أي (النقاط الثلاثة) يمكن أن تقبل صورة مسجلة بدلا من السابقة وهذه ميزة تضاف كونها قادرة على ترقيم صورة طبيعية وأن تكون لهذه الصورة شدة ضوئية كافية لإشارة المكشاف .

3- الحواسيب الرقمية و أجهزة الخزن

أن أنظمة الحواسيب المستعملة من أجل معالجة الصورة تتدرج من أجهزة المعالجات الصغيرة إلى أنظمة الحواسيب الضخمة القادرة على أنجاز دوال معقدة حسابيا على مصفوفات حساب كبيرة . أن المعلومات الأساسية التي تؤثر على بنية الحاسب المخصص لمعالجة الصور هي التطبيق المقصود وكمية البيانات المراد إدخالها وإخراجها من أجل التطبيق لهدف معين .

كيف نفرق بين الحاسوب الرقمية والتناظرية : الحاسوب التي نستخدمها في مختبراتنا وحاسباتنا الشخصية هي رقمية أما الحاسبات التناظرية فهي الحاسبات التي تستخدم لمعرفة الزلازل والبراكين والرياح التي تستخدم في الشفرات الجوية والحاسبات الهجينة تجمع بين النوعين الرقمي والتناظري .

أن صورة رقمية تتألف من 512*512 عنصر يمكن تهيئتها 8 بتأت (1 Byte) تتطلب 0.25 ميكابايت من التخزين أن وسائط التخزين هنا يمكن تقسيمها إلى ثلاثة أنواع :-

1- الأقراص المغناطيسية
2- الأشرطة المغناطيسية
3- الأقراص البصرية ال Optical

1- الأقراص المغناطيسية : تكون سعة 700 MB أو أكثر هي الشائعة حيث يمكنها الاحتفاظ ب2800 صورة من الحجم 512*512 .

2- الأشرطة المغناطيسية : عالية الكثافة من جهتيها (two side) 6400 بايت للانج الواحد تستطيع أن تخزن صورة واحده من الحجم 512*512 في أربعة أقدام تقريبا من الشريط .

3- الأقراص البصرية : تعتمد على تقنية الليزر في القراءة والكتابة أصبحت الآن متوفرة تجاريا . أن سعة التخزين لقرص كبير منفرد تصل إلى GB4 أي إلى 4000 مليون بايت في القرص الواحد .

مثال// بين كيفية أن الصورة حجمها 512 * 512 بكسل تتطلب 0.25 من التخزين

الحل//

$$512 * 512 = 2^{18} \;,\; M = 2^{20}$$
$$512 * 512 / 2^{20} = 2^{18} / 2^{20} = 0.25$$

أو طريقة أخرى

$$700\ MB / 2800\ MB = 0.25$$

4-أجهزة التسجيل والإظهار" العرض "

أن شاشات المراقبة التلفزيونية الأبيض والأسود والملون هي أجهزة الإظهار الرئيسية المستعملة في أنظمة المعالجات الصورية (معالجة صور) الحديثة .
أنظمة أنابيب الأشعة ألمهبطيه يتم فيها تحويل الموقعين الأفقي والعمودي لكل عنصر في الصورة إلى جهود تستعمل لحرف شعاع أنبوب الأشعة ألمهبطيه مؤمنة بذلك تحديده على شكل أبعاد ثنائية لإنتاج الصورة المخرجة .

أن أجهزة أظهار الطباعة مفيدة بشكل أساسي من أجل الأعمال المتعلقة لمعالجة الصورة بدقة منخفضة ،وهنالك أجهزة أخرى تتضمن طابعات الليزر وأجهزة الورق الحساس للحرارة وأجهزة رش الحبر.

5- تحصيل الصورة
يكون دخل تحصيل الصورة عبارة عن تحصيل صورة إشارة معينة (أشارة صوتية) أن أغلب معالجات الصورة الحديثة تقدر بزمن أطار واحد هو (1/30 Per second) من الثانية ولهذا السبب غالبا ما يشار إلى وحده تحصيل الصورة بقانص الإطار (Frame grabber) لأن كل صورة لها زمن معين 1/30 بالثانية وكذلك لها إطار حجم معين للصورة لذلك وجد قانص الإطار .

أن وحده التخزين التي غالبا ما تستدعى وتخزن قانص الإطار هي عبارة عن ذاكرة قادرة على تخزين صورة رقمية كاملة حيث يتم أعادة بناء وحدات هذا النوع من المعالج ولها القدرة على التخزين أيضا في نفس الوقت 30 صورة في الثانية أن هذه الخاصية تسمح بالسرعة بعملية التخزين .

أن وحده المعالجة تنجز الوظائف المنخفضة المستوى مثل العمليات الحسابية والمنطقية في وحده الحساب والمنطق (ALU) حيث تسمح المكونات هنا بالمعالجة بالشكل المتوازي (يعني أي خطاء لا يؤثر على بقية الأخطاء لأنه متوازي) .

أن وحده أو جهاز العرض يقوم بقراءة الصورة المخزونة بالذاكرة وتحويل المعلومات الرقمية المخزونة إلى أشارة تماثلية مرئية وإخراج هذه الإشارة إلى شاشة الجهاز لكي ترى من قبل المستخدم .

(بيانات مدخلة تناظرية ← معالجة رقمية (أرقام) ← أشارات تناظرية (المخرجات))

3-2 موديل تحليل الصورة

معالجة تحليل الصورة يمكن أن ندونها بالمراحل التالية :

1- المعالجة الابتدائية Pre processing

تستخدم هذه المعالجة لتحديد الضوضاء (النقط والنمش والريش) والمعلومات المرئية التي لا علاقة لها أو لا تؤثر على نتائج المناطق التي سوف تعالج لاحقا .

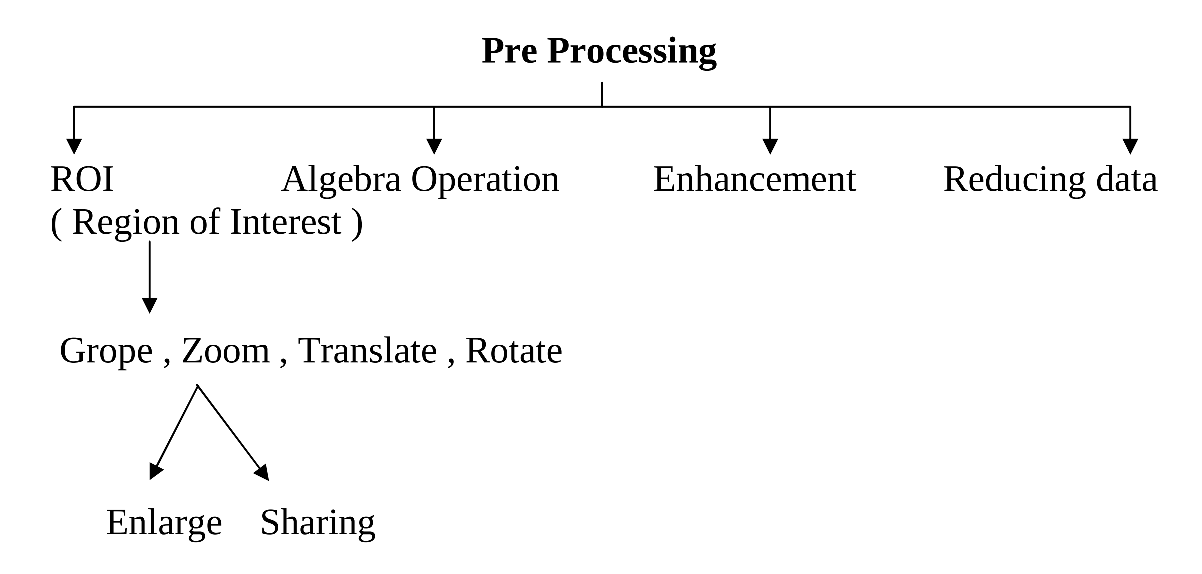
2- تقليل البيانات Data Reduction

وهي المرحلة التي تستخدم لتقليل البيانات في المجال ألحيزي وتنتقل النتيجة إلى مكان أخر يسمى المجال الترددي ونحدد الخواص (مجال ترددي ـــ ومجال حيزي)
لمعالجة التحليل نستخدم ألحيزي لأنة أسهل

3- تحليل الخواص المستخلصة Feature analysis

نستخدم هذه المرحلة باستخدام الخواص المستخلصة في المرحلة السابقة حيث تختبر وتقيم باستخدام أحدى التطبيقات

يمكن توضيح المعالجة الابتدائية بالشكل الأتي:



شكل(6): المعالجة الابتدائية

المعالجة الابتدائية تقسم إلى أقسام

1- هندسة الصورة لمنطقة داخلية أو نسخة داخلية معينة ك سوف نستخدم الخواص المستخلصة لمنطقة معينة تسمى (ROI) يتم هنا استخدام عمليات معينة يتم من خلالها تعديلها عن طريق إحداثيات حيزيه مستخدمة عمليات هندسة الصورة ومن هذه العمليات

Group أو Zoom أو توسيع أو تقليص أو نقل تدوير وبعد ذلك يتم الحصول على صورة جزئية نقوم بالمعالجة اللاحقة لها .

طريقة التكبير أو التصغير (Zoom Process) :

1- هناك طرق لمعالجة ال Zoom وأول طريقة هي طريقة الترتيب الصفري (Zero-Order-Hold) والتـي تـتم مـن خـلال إعـادة رقـم البكـسلات الـسابقة بتكـرار قـيم الصفوف والأعمدة مثلا نظيف صف صف لتكبير الصفوف أو إضافة أعمدة أو إضافة صفوف وأعمدة بنفس الوقت لتكبير المصفوفة أو العمود .

مثال// لديك جزء الصورة التالي المطلوب

1- تكبيرها بطريقة Zero-Order-Hold صف صف
2- تكبيرها بطريقة Zero-Order-Hold عمود عمود
3- تكبيرها بطريقة Zero-Order-Hold صف وعمود

40 20 10
70 50 30
90 80 10

الحل //

1- يكون الناتج مصفوفة سعتها 6×3

40	40	20	20	10	10
70	70	50	50	30	30
90	90	80	80	10	10

2- يكون الناتج مصفوفة سعتها 3×6

40	20	10
40	20	10
70	50	30
70	50	30
90	80	10
90	80	10

4- يكون الناتج مصفوفة سعتها 6×6

40	40	20	20	10	10
40	40	20	20	10	10
70	70	50	50	30	30
70	70	50	50	30	30
90	90	80	80	10	10
90	90	80	80	10	10

2- طريقة إيجاد المعدل

أيجاد المعدل بين قيمتين بكسلين متجاورين ووضع القيمة بينهما مثل 8 ، 4 نجمعها =12 نقسم على 2 فتصبح القيمة الوسطية هي 6 تكتب النتيجة

8 6 4

وإذا استخدمنا هذه الطريقة بطريقة معدل بكسلات الصفوف فسوف تزداد الأعمدة وإذا استخدمنا الأعمدة تزداد الصفوف .

3- يمكن أن نعمل على زوجي بكسل في كل صف وكل عمود وكما يمكننا أن نوسع الأعمدة والصفوف سوية .

هذه الطريقة تكبير سعه المصفوفة الN*N لتصبح مصفوفة صورة حجمها (2n-1*2n-1)

.

مثال//

إذا كانت لدينا المصفوفة هي 3*3 تمثل جزء من قيم الصورة الرقمية والمطلوب توسيع الأعمدة والصفوف سوية .

الحل // يصبح حجم المصفوفة 5*5

مثال للتوضيح // لدينا المصفوفة التالية سنجري عليها عمليات التوسيع للصفوف والأعمدة سوية

$$\begin{bmatrix} 8 & 4 & 8 \\ 4 & 8 & 4 \\ 8 & 2 & 8 \end{bmatrix} \xrightarrow{\text{توسيع}} \begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 4 & 6 & 8 & 6 & 4 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

يتم العمل على الصفوف والأعمدة على المصفوفة الناتجة وليس الأصلية

$$\Rightarrow \begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 6 & 6 & 6 & 6 & 6 \\ 4 & 6 & 8 & 6 & 4 \\ 6 & 11/2 & 5 & 11/2 & 4 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

3- طريقة التلفيف (الطي)

توجد طريقة أخرى جديدة تعطينا نفس النتائج بطريقة معالجة رياضية تسمى التلفيف (convolution) تتكون هذه الطريقة من خطوتين معالجة

1- توسيع الصورة بإضافة صفوف وأعمدة من الاصفار بين عناصر الصورة الأصلية (صفوف وأعمدة الصورة)

2- نبدأ بعملية التلفيف (لكل طريقة تلفيف تكبير أو تحسين لها مصفوفة خاصة بها (المصفوفة تتغير حسب نوع التغير)

مثال // لديك المصفوفة التالية التي تمثل جزء من الصورة المطلوب توسيع هذه الصورة بطريقة التلفيف .
الحل // 1- نضيف أعمدة أصفار وصفوف أصفار لتصبح المصفوفة على الشكل التالي

3	5	7
2	7	6
3	4	9

أصبحت المصفوفة 7 * 7

0	0	0	0	0	0	0
0	3	0	5	0	7	0
0	0	0	0	0	0	0
0	2	0	7	0	6	0
0	0	0	0	0	0	0
0	3	0	4	0	9	0
0	0	0	0	0	0	0

2- نقوم بعملية التلفيف أي نستخدم ماسك (Mask) قناع للتلفيف الذي يؤدي العملية الرياضية هذه لكل موقع بكسل (يجب أن يكون الماسك أو القناع أو النافذة بحجم المصفوفة الأصلية يعني مصفوفة تضرب بالمصفوفة التي عندنا ولكل عملية هنالك قناع ماسك خاص بها .

وقناع أو ماسك الWindow بماسك التلفيف الخاص بالتكبير هو ذو حجم 3*3

1/4	1/2	1/4
1/2	1	1/2
1/4	1/2	1/4

نأخذ من المصفوفة ما يساوي حجم الماسك (جزء من المصفوفة وليكن 3*3 وهو الجزء الأول فيها) نضرب المصفوفة ونستخرج الناتج مثلا نضرب الصف الأول من جزء المصفوفة المختار (المصفوفة 3*3 المختارة) ونضرب بما يقابلها في الماسك

$$1/4*0 + 1/2*0 + 1/4*0 + 1/2*0 + 1*3 + 1/2*0 + 1/4*0 + 1/2*0 + 1/4*0 = 3$$

فتوضع القيمة الناتجة من الضرب وهي 3 في أول موقع للمصفوفة الجديدة
وتبقى في نفس الصف (الصف الأول من المصفوفة الكبيرة ولكن هنا نزحف المصفوفة المختارة (جزء من المصفوفة الكبيرة) لعمود واحد فقط أي العمود الثاني في الصف الأول من المصفوفة الكبيرة ونستمر هكذا إلى أن ننتهي كل أعمدة الصف الأول ثم فننزل إلى الصف الثاني وأيضا نتبع نفس الطريقة ألطريقه في عملية تزحيف الأعمدة .

نضرب هذا الجزء بمصفوفة الماسك

0 0 0 0
0 3 0 5
0 0 0 0

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 3*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=3

توضع في المصفوفة الجديدة القيمة الأول

$$\begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 5 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 3*1/2 + 0*1 + 5*1/2 + 0*1/4 + 0*1/2 + 0*1/4=4

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0 1/2 + 0*1/4 + 0*1/2 + 5*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4 = 5

$$\begin{bmatrix} 0 & 0 & 0 \\ 5 & 0 & 7 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 5*1/2 + 0*1 + 7*1/2 + 0*1/4 + 0*1/2 + 0*1/4 =6

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 7*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=7

ثم تنزل للصف الثاني ونتبع نفس الطريقة بالتزحيف

$$\begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 3*1/2 + 0*1/4 + 0*1/2 + 0*1 + 0*1/2 + 0*1/4+2*1/2+ 0*1/4=5/2

$$\begin{bmatrix} 3 & 0 & 5 \\ 0 & 0 & 0 \\ 2 & 0 & 7 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

3*1/4 + 0*1/2 + 5*1/4 + 0*1/2 + 0*1 + 0*1/2+2*1/4+0*1/2+ 7*1/4=17/4

$$\begin{bmatrix} 0 & 5 & 0 \\ 0 & 0 & 0 \\ 0 & 7 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 5*1/2 + 0*1/4 + 0*1/2 + 0*1 + 0*1/2+0*1/4+7*1/2+ 0*1/4=6

$$\begin{bmatrix} 5 & 0 & 7 \\ 0 & 0 & 0 \\ 7 & 0 & 6 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

5*1/4 + 0*1/2 + 7*1/4 + 0*1/2 + 0*1 + 0*1/2+7*1/4+0*1/2+ 6*1/4=25/4

$$\begin{bmatrix} 0 & 7 & 0 \\ 0 & 0 & 0 \\ 0 & 6 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 7*1/2 + 0*1/4 + 0*1/2 + 0*1 + 0*1/2+0*1/4+6*1/2+ 0*1/4=13/2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 2*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=2

$$\begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 7 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 2*1/2 + 0*1 +7*1/2 + 0*1/4 +0*1/2+ 0*1/4=9/2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 7*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=7

$$\begin{bmatrix} 0 & 0 & 0 \\ 7 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 7*1/2 + 0*1+ 6*1/2+0*1/4+ 0*1/2+ 0*1/4=13/2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 6*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=6

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 2*1/2 + 0*1/4 + 0*1/2 + 0*1 + 0*1/2 + 0*1/4+3*1/2+ 0*1/4=5/2

$$\begin{bmatrix} 2 & 0 & 7 \\ 0 & 0 & 0 \\ 3 & 0 & 4 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

2*1/4 + 0*1/2 + 7*1/4 + 0*1/2 + 0*1+ 0*1/2+ 3*1/4+ 0*1/2+ 4*1/4=4

$$\begin{bmatrix} 0 & 7 & 0 \\ 0 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 7*1/2 + 0*1/4 + 0*1/2+ 0*1+ 0*1/2+ 0*1/4+4*1/2 + 0*1/4=11/2

$$\begin{bmatrix} 7 & 0 & 6 \\ 0 & 0 & 0 \\ 4 & 0 & 9 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

7*1/4 + 0*1/2 + 6*1/4 + 0*1/2 + 0*1+0*1/2+4*1/4+0*1/2 + 9*1/4=26/4

$$\begin{bmatrix} 0 & 6 & 0 \\ 0 & 0 & 0 \\ 0 & 9 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 6*1/2 + 0*1/4 + 0*1/2 + 0*1+0*1/2+0*1/4+9*1/2 + 0*1/4=15/2

ثم ننزل إلى الصف الأخير بالمصفوفة الكبيرة

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 3*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=3

$$\begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 3*1/2 + 0*1 + 4*1/2 +0*1/4+0*1/2+ 0*1/4=7/2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 4*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=4

$$\begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 9 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 4*1/2 + 0*1 + 9*1/2+0*1/4+0*1/2+ 0*1/4=13/2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} =$$

0*1/4 + 0*1/2 + 0*1/4 + 0*1/2 + 9*1 + 0*1/2 + 0*1/4 + 0*1/2 + 0*1/4=9

المصفوفة الناتجة بعد عملية التلفيف هي

3	4	5	6	7
17/4	6	25/4	13/2	5/2
9/2	7	13/2	6	2
4	11/2	26/4	15/2	5/2
7/2	4	13/2	9	3

عدد المرات التي استخدمنا فيها الماسك هي 25

يمكن أن نجد قاعدة لإيجاد المعدل بين الجيران إذا افترضنا بأن ماسك التلفيف هو M(r,c)
مصفوفة وقيم عناصر الصورة هي I(r,c)
فأننا نحصل على معادلة تلفيف هي

$$\sum_{x=\infty} \sum_{y=\infty} I (r - x , c - y) M (x , y) ....(3)$$

تحول الماسك إلى x ,y لأنة تحول إلى البكسلات r-x , c-y الناتج عن التزحيف للصفوف والأعمدة ؟ برهن ذلك ؟
الجواب // نقوم بفك ال∑ وال(M(x,y هي حجم المصفوفة وهي 3*3 وال r-x , c-y هي المصفوفة بعد التوسيع وهي r,c وهي 7*7 والM=9 يعني حجم المصفوفة هي 7*7 والx والy هي قيم ال∑ التي تبدأ من 1 إلى 3

X	Y	I(r-x, c-y) *M (x, y)
1	1	I(6 , 6) * M(1 , 1)
1	2	I(6 , 5) * M(1 , 2)
1	3	I(6 , 4) * M(1 , 3)
2	1	I(5 , 6) * M(2 , 1)
2	2	I(5 , 5) * M(2 , 2)
2	3	I(5 , 4) * M(2 , 3)
3	1	I(4 , 6) * M(3 , 1)
3	2	I(4 , 5) * M(3 , 2)
3	3	I(4 , 4) * M(3 , 3)

مثال// لماذا تتطلب التلفيف طريقة العديد من الحسابات بالمقارنة مع طريقة أيجاد المعدل للجيران ؟ ما هو السبيل لحل هذه المشكلة ؟

الجواب // سعة المصفوفة الأصلية يكون أكبر بالنسبة للتلفيف مثلا I(3*3) تصبح I(7*7) بينما بالمعدل فأن سعة المصفوفة I(3*3) تصبح I(5*5) يعني التلفيف تكبير السعة وبهذه الطريقة نعتمد على الماسك والطريقة بالمعدل لا تعتمد على الماسك بل فقط على عناصر الصورة ونجد المعدل بين العنصرين فقط .

بالتلفيف : 1- حجم الصورة سعتها كبيرة
2- تعتمد على الماسك الذي يضرب بالعناصر .
والحل لهذه المشكلة هو استخدام الحاسوب لأجراء عملية التلفيف حيث تسهل وتسرع الخوارزمية .

يمكن أجراء الترتيب الأولي (First order) باستخدام التلفيف بحيث يمثل طريقة الZero ويساعد على التلفيف (توسيع) المصفوفة (الصورة) مع قيم الاصفار باستخدام ماسك ثاني هو

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

وتتضمن هذه الطريقة كالأتي :

1- حشر صفوف وأعمدة من الاصفار (نفس الطريقة السابقة)
2- نأخذ الماسك بحجم (2*2) من الصورة ونقوم بأجراء عملية الضرب والجمع لكن هنا يجب أن تعطى النتائج في الجهة اليمنى السفلى .

$$\begin{bmatrix} 3 & 2 & 1 \\ 4 & 9 & 5 \\ 6 & 7 & 6 \end{bmatrix}$$

وهذه المصفوفة تصبح بعد التلفيف

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 7 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

بعد ضرب المصفوفة بالماسك أعلاه تكون المصفوفة النهائية هي

$$\begin{matrix} 3 & 3 & 2 & 2 & 1 & 1 \\ 3 & 2 & 2 & 1 & 1 & 3 \\ 4 & 9 & 9 & 5 & 5 & 4 \\ 4 & 9 & 9 & 5 & 5 & 4 \\ 6 & 7 & 7 & 2 & 2 & 6 \end{matrix}$$

6 7 7 2 2 6

هذه النتيجة بالZero هي تشبه أو نفس طريقة التلفيف بالاصفار عندما تضرب بالماسك

4- التكبير باستخدام المعامل K :

يعني مثلا كبر الصورة (المصفوفة) مثلا 3 مرات بحجمها يعني المعامل K = 3 يضرب في سعة المصفوفة .

أذا كان المطلوب تكبير مصفوفة (جزء من صورة) تكبيرها ثلاث أو أربع مرات أو غيرها ... نستخدم ما يسمى بمعامل K ونقوم بالتالي :

1- طرح قيمة كل قيمتين متجاورتين .
2- قسمة الناتج على معامل التكبير K
3- إضافة النتيجة إلى أصغر قيمة ونستمر بالإضافة لكل العناصر بمقدار K-1 .
4- تطبيق هذه الخطوات على الصفوف والأعمدة .

مثال // لديك جزء الصورة التالية المطلوب تكبيرها بمقدار 3 مرات من حجمها الأصلي ؟

$$\begin{bmatrix} 125 & 140 & 155 \end{bmatrix}$$

الحل // نأخذ كل قيمتين متجاورتين فنطرح الصغير من الكبير

140 – 125 = 15

ثم نقسم الناتج على 3   15/3 = 5

ثم نضيف نتيجة القسمة إلى أصغر قيمة فتصبح   125 + 5 = 130

K = K – 1 = 3 – 1 = 2

أي نضيف 5 مرتين   130 + 5 = 135

فتصبح النتيجة هناك _ رقمين بين ال125 وال140

$$\begin{bmatrix} 125 & 130 & 135 & 140 & .. \end{bmatrix}$$

ثم نأخذ الرقمين المتجاورين الأخريين وهما 140 و155

155 – 140 = 15

15/3 = 5

K = K – 1 = 3 – 1 = 2

140 + 5 = 145

145 + 5 = 150

فتصبح المصفوفة على الشكل التالي

$$\begin{bmatrix} 125 & 130 & 135 & 140 & 145 & 150 & 155 \end{bmatrix}$$

واجب // لديك المصفوفة التالية المطلوب تكبيرها أربع مرات K = 4 ؟

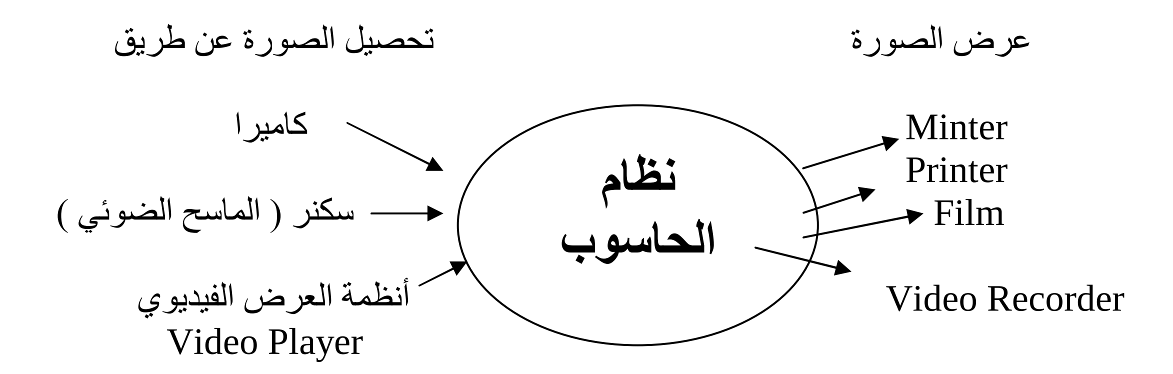
$$\begin{bmatrix} 140 & 160 & 180 \\ 160 & 180 & 200 \end{bmatrix}$$

4-2 أنظمة صور الحاسوب

يمكن أن تقسم المكونات الابتدائية للحاسب إلى المكونات المادية والبرمجيات :

1- المكونـات الماديـة : يمكـن أن تقسم إلـى الأنظمـة الفرعيـة لتحصيل الصورة أو الحاسوب نفسها واستخدام أجهزة العرض .
2- البرمجيات : يمكن تمثيلها كوصف للبرامج المستخدمة لتمثيل بيانات الصورة ويمكن لهذه البرمجيات السيطرة على تحصيل الصورة ومعالجة الخزن .

H/W : تحصيل الصورة والعرض والخزن
S/W : البرامج (البيانات التناظرية تحول إلى بيانات رقمية)



الشكل(7): يمثل المكونات المادية لنظام صورة بالحاسوب

نظام الحاسوب يمكن أن يستخدم كنظام متعدد الأغراض من خلال ما يسمى بقانص الإطار أو المرقم للصورة كوحدة للإدخال للصورة المستخدمة . يمكن أن نعتبر قانص الإطار كغرض خاص بالمكونات المادية لانه يقبل الإشارة الفيديوية القياسية وإخراج الصورة بصيغة معينة من خلال الحاسوب بحيث يفهمها المستخدم.

قانص الصور ← غرض عام
قانص الصور ← غرض خاص : استخدام جزء من الصورة

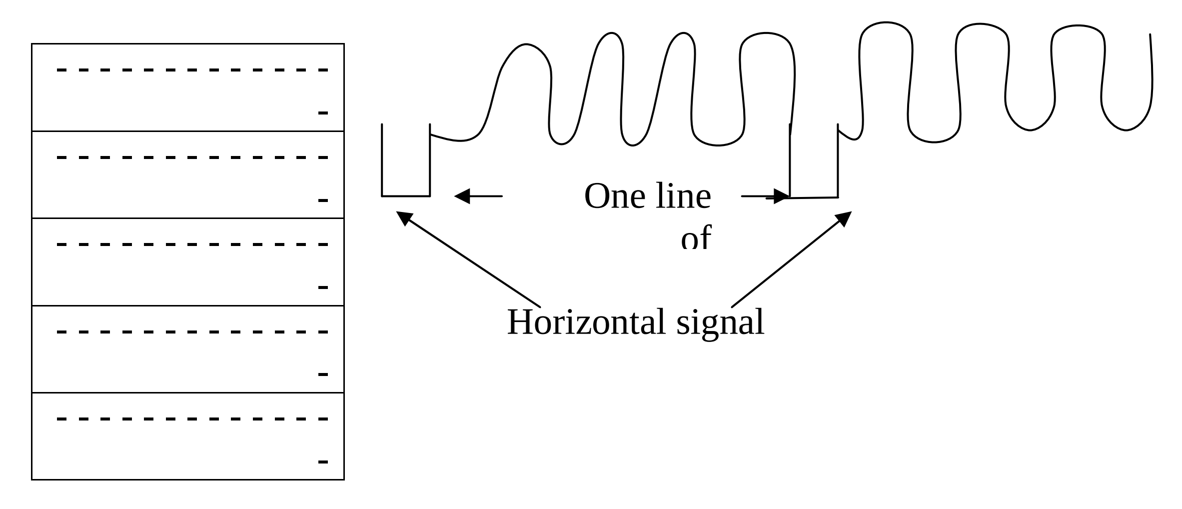
إشارة فيديوية قياسية ← يستقبل الصورة ← القانص ← الغرض الخاص لقانص الإطار

← الإخراج (بعد التحويل بصيغه يفهمها المبرمج) .

الغرض العام : يستخدم الصورة كاملة بحيث يقوم قانص الإطار كمرقم للصورة الكاملة فالصورة الرقمية هنا هي الصورة المحولة من قبل قانص الإطار إلى صورة رقمية .

الترقيم (Digitalization) : وهي معالجة تحويل الصورة من الإشارة الفيديوية القياسية إلى صورة رقمية فالتحويل هذا ضروري لأن الإشارة الفيديوية القياسية تكون بصيغة تناظرية (Continuous مستمرة) لذا يتطلب التحويل إلى صيغة أخرى تسمى الإشارة الرقمية (Symbol عينات) أو أرقام .

الإشارة الفيديوية تتكون من مجموعة من الإطارات الفيديوية التي تحتوي على المعلومات ، كل إطار يتكون من مكونات تملأ الشاشة بصيغة مرئية للمعلومات أي أنة يمكن أن نقسم هذا الإطار إلى حقول وهذه الحقول بدورها تحتوي على خطوط خاصة بالمعلومات الفيديوية .



a- one frame **b- video signal**

شكل(8): الإشارة الفيديوية

5-2 صور الحاسوب

يمكن أن نعرفها بأنها تحصيل (اكتساب) ومعالجة المعلومات المرئية بواسطة الحاسوب و يمكن أن نقسمها إلى مستويين مختلفين من المساحة هما :

1- **الرؤية بالحاسوب**
2- **معالجة الصور**

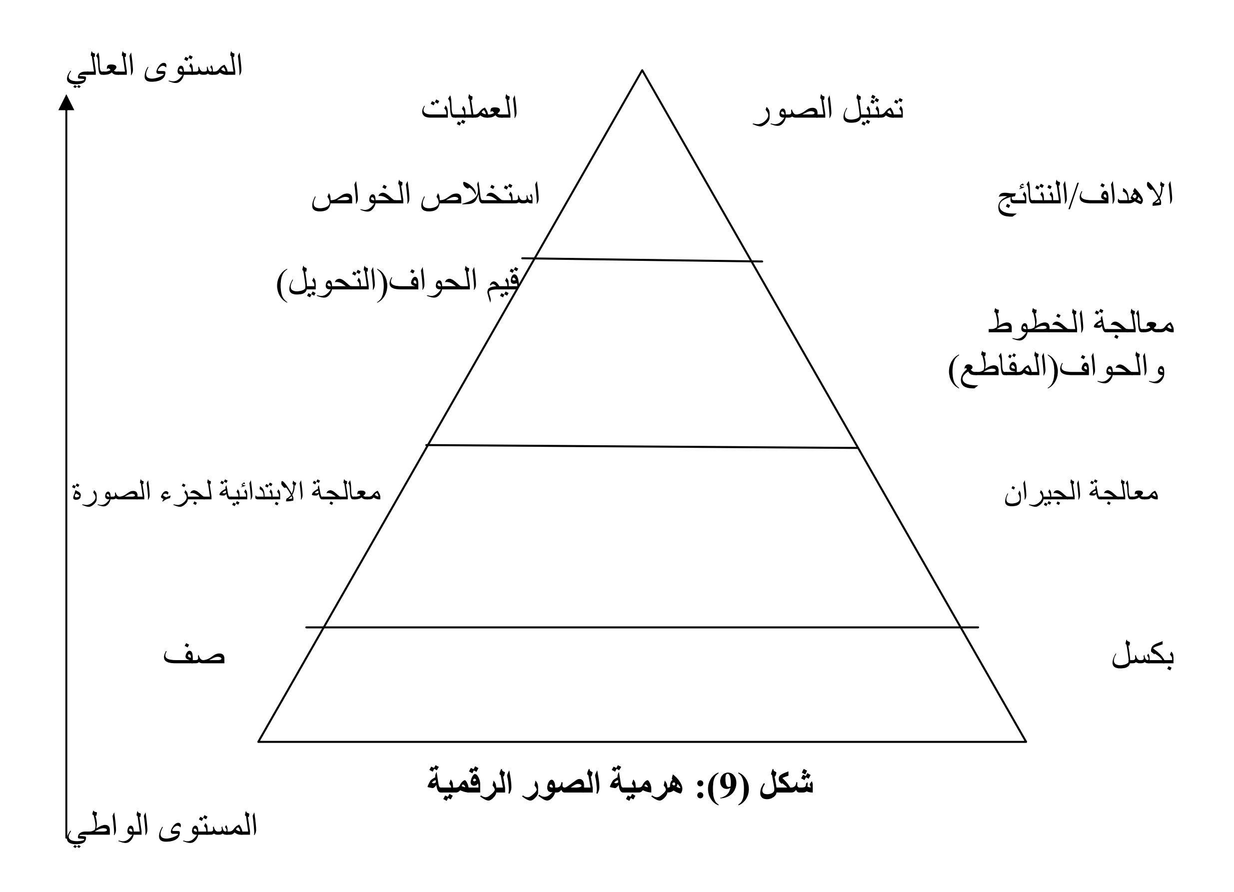
الرؤية بالحاسوب : هي صور الحاسوب التي يكون فيها التطبيق لا يستخدم من خلال الشخص كبداية في عمليات تدويريه تكرارية مرئية للصورة المختبرة وفعاليتها بواسطة الحاسوب وهي واحدة من الخصائص أو الحقول التي نمثلها في الرؤية بالحاسوب مثل تحليل الصورة حيث يستخدم تحليل الصورة لاختيار بيانات الصورة الخاصة بعمليات بيانات الصور والتسهيل وحلول الرؤية بالحاسوب يمكن أن نقسمها بصورة عامة إلى قسمين

1- استخلاص الخواص
2- تصنيف النماذج

استخلاص الخواص : هو معالجة معلومات الصورة ذات المستوى العالي المكتسبة مثل الحدود والألوان للصورة وغيرها .

تصنيف النماذج : هو الفعالية المستخدمة لمعالجة معلومات المستوى العليا للصورة وتعريف الأهداف الخاصة بها .

معالجة الصورة : هي المعالجة التي يكون فيها تطبيق الصورة معتمدا على الشخص (العمليات التكرارية المرئية الممتحنة والفعالة للشخص) ويمكن أن نقسم معالجة الصورة إلى أربعة أنواع رئيسية هي :



شكل (9): هرمية الصور الرقمية

ملاحظة// شكل المثلث يمثل هرمية الصورة .

المستوى الأول/ المتطلبات بالبكسل (تعالج البيانات بشكل خطي لبيانات صف الصورة لأن المعالجة هي صف صف

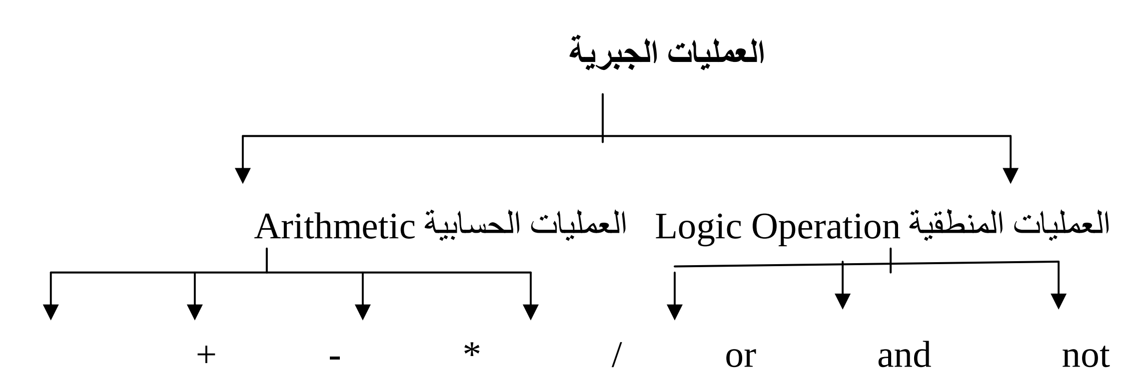
المستوى الثاني/ جيران الكسل (صورة جزئية) سوف نعمل لها معالجة ابتدائية تكبير أو تصغير)

المستوى الثالث /(خطوط وحواف مقطعة طيفية) نعمل لها عمليات التحويل والتقطيع وتحديد الحواف

المستوى الرابع الأخير/ (خواص الأهداف يعني نعمل استخلاص الخواص) .

6-2 العمليات الجبرية Image Algebra

العمليات الجبرية تقسم إلى عمليات رياضية (عمليات حسابية) وعمليات منطقية ويمكن تحديد العمليات الجبرية بالمخطط التالي



شكل(10):مخطط يوضح العمليات الجبرية للصورة الرقمية

العمليات الحسابية:

*** عملية الجمع (Addition)**

عملية الجمع تستخدم لجمع معلومات صورتين من خلال جمع عناصر الصورة الأولى مع الثانية مبتدئين بالعنصر الأول من الصورة الأولى مع العنصر الأول من الصورة الثانية وهكذا بالنسبة لبقية العناصر . ونستخدم طريقة الجمع في أعادة أو ترقيم الصورة Image Restoration و لإضافة الضوضاء للصورة Noise (كنوع من أنواع التشفير) .

مثال // لديك جزئي الصورتين التاليتين الصورة الأولى هي I_1 والصورة الثانية هي I_2 المطلوب جمع هاتين جمع الجزائين؟

$$I_1 \begin{bmatrix} 3 & 4 & 7 \\ 2 & 4 & 5 \\ 2 & 4 & 6 \end{bmatrix} \qquad I_2 \begin{bmatrix} 6 & 6 & 6 \\ 4 & 2 & 6 \\ 3 & 5 & 5 \end{bmatrix}$$

الحل //

$$I_1 \begin{bmatrix} 3 & 4 & 7 \\ 2 & 4 & 5 \\ 2 & 4 & 6 \end{bmatrix} + I_2 \begin{bmatrix} 6 & 6 & 6 \\ 4 & 2 & 6 \\ 3 & 5 & 5 \end{bmatrix}$$

$$=I3=\begin{bmatrix} 9 & 10 & 13 \\ 6 & 6 & 11 \\ 5 & 9 & 11 \end{bmatrix}$$



شكل(11):تستخدم الجمع لزيادة إضاءة صورة

مثال // في حالة استخدام الجمع للـNoise ما هي الطريقة برأيك لإعادة الصورتين ؟
الجواب // سوف نعتمد على النتيجة حيث نطرح إحدى المصفوفتين من النتيجة فالمصفوفة الناتجة من الطرح هي مصفوفة الضوضاء حيث نبدأ من الهدف ونخترق احتمالات الهدف باستخدام احتمالات فضاء البحث

*** عملية الطرح :**

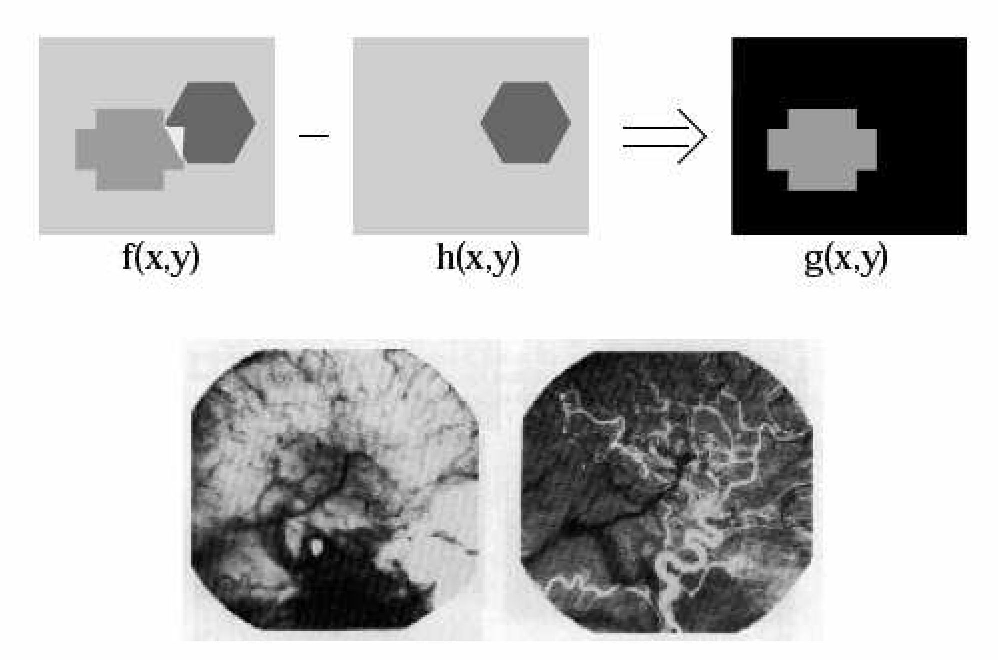
تستخدم عملية الطرح لطرح معلومات صورتين بحيث نطرح كل عنصر في الصورة الأولى مع العنصر الذي يقابله من الصورة الثانية . حيث يستخدم الطرح في تحديد الحركة على افتراض أن الصورتين متشابهتين لكن الأشياء الموجودة داخل الصورة مختلفة .
مثال // لديك الصورتين الآتيتين المطلوب طرح الصورتين ؟؟

$$I_1\begin{bmatrix} 7 & 3 & 2 \\ 9 & 8 & 6 \\ 3 & 3 & 3 \end{bmatrix} \qquad I_2\begin{bmatrix} 6 & 1 & 1 \\ 5 & 3 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

الحل //

$$I_1\begin{bmatrix}7 & 3 & 2\\9 & 8 & 6\\3 & 3 & 3\end{bmatrix} + I_2\begin{bmatrix}6 & 1 & 1\\5 & 3 & 2\\1 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\\4 & 5 & 4\\2 & 2 & 2\end{bmatrix}$$

في عملية الطرح المتشابه سوف يحذف والمختلف سوف يبقى ويمكن تحديد الحركة بسهولة



شكل(12):عملية الطرح للصورة

*** عملية الضرب**

تتم العملية بضرب عناصر المصفوفة الخاصة بالصورة بمعامل أكبر من الواحد وتستخدم لزيادة أو تقليص الصورة

مثلا : المعامل K يجب أن يكون أكبر من الواحد عندما نريد تكبير الصورة

مثال // لديك الصورة التالية المطلوب زيادتها وتقليصها بإحدى عمليات الجبرية للصور الرقمية

$$\begin{bmatrix}3 & 4 & 7\\3 & 4 & 5\\2 & 4 & 6\end{bmatrix}$$

الجواب //

باستخدام عملية الضرب 1- زيادتها 2- تقليصها

نستخدم الضرب كعمليات حسابية جبرية مثلا تضرب المصفوفة هذه في معامل (هنا نحن نختار المعامل لأن لم يحدد لنا بالسؤال)

يكون المعامل هو K=3 أكبر من الواحد في حالة الزيادة

$$\begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 5 \\ 2 & 4 & 6 \end{bmatrix} * 3 = \begin{bmatrix} 9 & 12 & 21 \\ 9 & 12 & 15 \\ 6 & 12 & 18 \end{bmatrix}$$

في حالة تقليص الصورة تضرب في 3- أقل من الواحد

$$\begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 5 \\ 2 & 4 & 6 \end{bmatrix} * -3 = \begin{bmatrix} -9 & -12 & -21 \\ -9 & -12 & -15 \\ -6 & -12 & -18 \end{bmatrix}$$

مثال// مثلا لدينا المصفوفة (جزء من صورة) التي تعتبر ناتج عملية الزيادة والـ(K=3)
المطلوب أيجاد المصفوفة الأصلية ؟؟
الجواب // هناك طريقتين للحل

1- نقسم المصفوفة على معامل K حيث تقسم كل قيمة موجودة بالمصفوفة الناتجة على المعامل (3) فتنتج المصفوفة الأصلية .
2- بدلالة النقصان بحيث نستخدم المعامل K أقل من الواحد مثلا K=1/3

ملاحظة//
$K > 1$ في حالة الزيادة يجعل الصورة تميل للبياض
$K < 1$ في حالة النقصان (التقليص) وهنا الصورة تميل للسواد (الظلام)

*** عملية القسمة**

تقسم عناصر الصورة المعطاة على معامل أكبر من الواحد حيث تجعل عملية القسمة هذه الصورة بشكل مظلم .

مثال // لديك المصفوفة التالية وهي جزء من صورة المطلوب تقسيم الصورة على معامل K=4

$$\begin{bmatrix} 8 & 12 & 16 \\ 8 & 16 & 8 \\ 20 & 24 & 16 \end{bmatrix}$$

الحل //

$$\begin{bmatrix} 8 & 12 & 16 \\ 8 & 16 & 8 \\ 20 & 24 & 16 \end{bmatrix} \div 4 = \begin{bmatrix} 2 & 3 & 4 \\ 2 & 4 & 2 \\ 5 & 6 & 4 \end{bmatrix}$$

إذا المعامل أكبر من الواحد يجعل الصورة بالقيم تميل إلى السواد على عكس عملية الضرب التي تجعل الصورة تميل للبياض . وذلك لأن الصورة قد صغرت أي قريبة للصفر وهو جهة السواد أما الضرب فهو زيادة لأنة يكبر ويقترب للجهة البياض .

العمليات المنطقية :

*** العملية المنطقية AND**

يتم تطبيق العمليات المنطقية على عناصر الصورة بعد تحويل كل عنصر من عناصر الصورة إلى الحالة الثنائية (Binary) بحيث يمكن استخدام العمليات المنطقية فيها من خلال طريقة (ROI) .
تعتبر الـAND شبيهه بعملية الضرب أي أن الصورة تميل إلى البياض وتتم من خلال مربع أبيض مع عناصر الصورة بحيث يكون الناتج جزء من الصورة المقابل للمربع الأبيض .

(AND : تجعل الجزء الذي نريده نجعل خلفيته بلون أبيض أما الـOR نجعل خلفيه الجزء المراد سوداء)

*** العملية المنطقية OR**

تتم هنا بأخذ مربع اسود وخلفية بيضاء لبيانات الصورة المطلوبة من الصورة الأصلية وأن عملية الـOR تشبه عملية الجمع .

*** العملية المنطقية NOT**

تستخدم بإعطاء القيم السالبة للصورة الأصلية أي يحدد عكس الصورة (مثل فلم الكاميرة negative)
أي تتم عكس بيانات الصورة أي الأسود يصبح أبيض والأبيض يصبح اسود

مثال // أذا كان لديك جزء الصورة التالي المطلوب استخدام الNOT؟

$$\begin{bmatrix} 7 & 3 & 2 \\ 9 & 8 & 6 \\ 3 & 3 & 3 \end{bmatrix}$$

الحل //

$$\begin{bmatrix} 7 & 3 & 2 \\ 9 & 8 & 6 \\ 3 & 3 & 3 \end{bmatrix} * -1 = \begin{bmatrix} -7 & -3 & -2 \\ -9 & -8 & -6 \\ -3 & -3 & -3 \end{bmatrix}$$

والصورة الناتجة من عملية الNOT هي قريبة للسواد وبيانات هذه الصورة يجب أن تحول إلى صيغة ال(Binary) (1 ، 0) .




شكل(13): عملية not للصورة الأصلية

مثال // طبق عملية بوابة ب(AND) لعنصر من الصورة بحيث العنصر الأول هو 88 والعنصر الثاني هو 111

الحل //

نحول ال88 إلى الصيغة الثنائية (1 ، 0) بحيث يكون

2	88	
2	44	0
2	22	0
2	11	0
2	5	1
2	2	1
2	1	0
2	0	1
		0

01011000

2	111	
2	55	1
2	27	1
2	13	1
2	6	1
2	3	0
2	1	1
2	0	1
		0

01101111

NOT (الANDوالORوالNOT) يجب أن تكون كلها Binary

01101111_2

AND * 01011000_2

01001000_2

01101111_2

OR + 01011000_2

11000111_2

أما في حالة الـ NOT فهي لأحد الرقمين بحيث يصبح كل صفر هو واحد وكل واحد هو صفر
الرقم الأول بعد عمليه الNOT 10010000 ← 01101111
الرقم الثاني بعد عملية الNOT 10100111 ← 01011000

ملاحظة// تكون القيم هنا مشوهه لذا استخدمت طريقة ثانية للتعامل مع هذه المعاملات بالنسبة للعمليات المنطقية وتحويلها إلى ثنائية وأيضا بالنسبة للبوابات
(NAND , NOR , XOR)

*** عملية النقل Transpose والتدوير Rotation**

1- عملية النقل Transpose

$$\bar{r} = r + r_0 \ldots\ldots\ldots\ldots\ldots\ldots(4)$$

$$\bar{c} = c + c0 \ldots\ldots\ldots\ldots\ldots\ldots(5)$$

مثال // النقطة (2 و 3) نريد نقلها مسافة $r_0 = 5$ و $c_0 = 5$
فتصبح (3+5 , 2+5) = (8 ، 7)

3- التدوير Rotation

$$r\hat{} = r(\cos\Theta) + c(\sin\Theta)$$
$$c\hat{} = -r(\sin\Theta) + c(\cos\Theta)$$

هنا نحتاج إلى الزاوية والنقاط ومقدار التدوير

وهنالك معادلة لدمج المعادلتين السابقتين للنقل والتدوير وهي :

$$R\bar{}\hat{} = (r + r_0)(\cos\Theta) + (c + c_0)(\sin\Theta) \ldots\ldots(5)$$
$$C\bar{}\hat{} = -(r + r_0)(\sin\Theta) + (c + c_0)(\cos\Theta) \ldots\ldots(6)$$

مثال// جزء المصفوفة التالي المطلوب نقل وتدوير ودمج النقل والتدوير حيث أن $r_0=5$ و $c_0=3$
والزاوية هي 90 والتكرار لثلاث مرات ؟

7	3	2
9	8	6
3	3	3

7-2 تحسين الصورة (المرشحات ألحيزيه) (Special Filter):

مرشح (Filter) يعني عملية تقوم بتصفية الصورة من الشوائب العالقة أي تبرز ملامح الجزء الذي نريده من الصورة بإزالة الضوضاء و الشوائب .

نستخدم المرشحات ألحيزيه لإزالة الضوضاء أو لتحسين الصورة حيث تطبق هذه المرشحات في مجال الصورة مباشرة (على عناصر الصورة مباشرة) وليس في مجال التردد (التحويل) حيث تستخدم عناصر الصورة باستخدام أحدى التحويلات مثل تحويل فورير وتحويلات الcos وتقسم الفلاتر إلى ثلاث أنواع :

1- مرشح المتوسط Mean Filter
2- مرشح الوسيط Median Filter
3- مرشح التحسين Enhancement Filter

نستخدم النوعين الأول والثاني لإزالة الضوضاء بالإضافة إلى بعض التطبيقات التي تعطي شكل التنعيم للصورة أي 1- إزالة الضوضاء 2- التنعيم
أما النوع الثالث يستخدم لتوضيح الحافات والتفاصيل الموجودة في الصورة حيث تطبق المرشحات ألحيزيه أما باستخدام العناصر مباشرة بدون استخدام ماسك أو عن طريق ماسك تلفيف مع العناصر ومجاوراتها.

يمكن معرفة نتائج الماسك كالتالي :

1- إذا كان مجموع معاملات الماسك يساوي 1 يعني إضاءة عالية للصورة .
2- إذا كان مجموع المعاملات يساوي 0 فأن إضاءة الصورة تفقد أي تميل للسواد
3- إذا كانت المعاملات سالبة وموجبة يعني ذلك معلومات عن الحواف .
4- إذا كانت المعاملات موجبة فقط يحصل نوع من التشويه بالصورة .

1- مرشح المتوسط Mean Filter

هو مرشح خطي عناصره هي

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

وكل عناصره موجبة ولأنها كلها موجبة إذا هنالك تشويه بالصورة
وبما أن مجموع عناصر الماسك يساوي 1 إذن هناك إضاءة عالية

مثال // طبق ماسك الMean على الصورة التالية

$$\begin{bmatrix} 2 & 3 & 5 & 4 \\ 1 & 5 & 6 & 7 \\ 2 & 3 & 9 & 11 \end{bmatrix}$$

الحل //

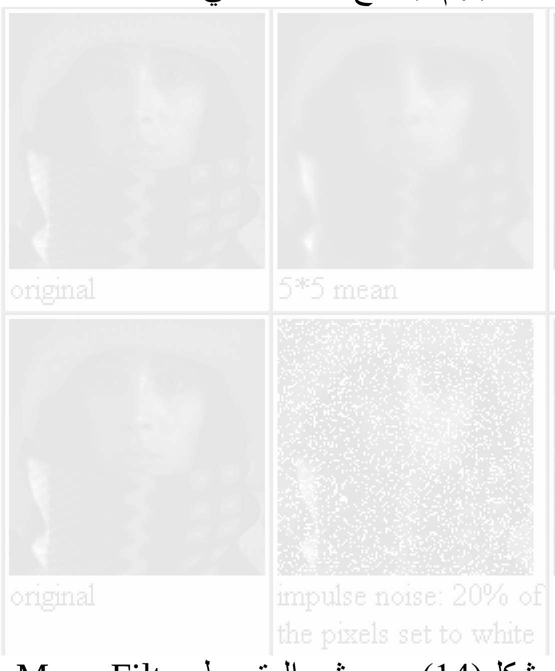
$$\begin{bmatrix} 2 & 3 & 5 \\ 1 & 5 & 6 \\ 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$2/9 + 3/9 + 5/9 + 1/9 + 5/9 + 6/9 + 2/9 + 3/9 + 9/9 = 4$$

$$\begin{bmatrix} 3 & 5 & 4 \\ 5 & 6 & 7 \\ 3 & 2 & 11 \end{bmatrix} * \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$3/9 + 5/9 + 4/9 + 5/9 + 6/9 + 7/9 + 3/9 + 2/9 + 11/9 = 6$$

وبنا أن الناتج نقطتين فنوصل بينهم فيصبح الشكل خطي



شكل(14): - مرشح المتوسط Mean Filter

2- مرشح الوسيط Median Filter

هو عبارة عن مرشح لا خطي يعمل على عناصر الصورة مباشرة بعد تحديد ماسك من خلال العناصر حيث يستبدل مركز الصورة بالقيمة التي في الوسط

مثال // طبق Median Filter على جزء الصورة التالية ؟

$$\begin{bmatrix} 5 & 5 & 6 \\ 3 & 4 & 5 \\ 3 & 4 & 7 \end{bmatrix}$$

لا يوجد ماسك بالMedian ونحن نعمل لها ماسك من عناصر المصفوفة حيث نأخذ عناصر الصورة ونعمل فيها ماسك وذلك بترتيب عناصر الصورة تصاعديا فتصبح :

1- الخطوة الأولى : نرتب العناصر تصاعديا

3 3 4 4 **5** 5 6 7

2- الخطوة الثانية : نقسم عدد العناصر على 2 لنستخرج الموقع الوسط

9/2 = 4.5 → 5

3- الخطوة الثالثة : نرى ما هي قيمة الموقع الخامس

قيمة الموقع الخامس 5

ليس شرطا أن تتساوى القيمة مع الموقع الوسيط

4- الخطوة الرابعة : نغير الموقع الوسط بالمصفوفة وهو الموقع الخامس (4) الذي يكون مساويا للقيمة 5 بدل العنصر الوسط بالمصفوفة الأصلية وهو 4 (أي نضع 5 بدل ال4) وتصبح

$$\begin{bmatrix} 5 & 5 & 6 \\ 3 & 5 & 5 \\ 3 & 4 & 7 \end{bmatrix}$$

مثال // لديك جزء الصورة التالية

$$\begin{bmatrix} 2 & 3 & 5 & 4 \\ 1 & 15 & 6 & 7 \\ 2 & 3 & 9 & 11 \end{bmatrix}$$

الحل //

1- نأخذ الجزء (3*3) ونرتبه تصاعديا

1 2 2 3 **3** 4 6 9 15

2- نحدد العنصر في الوسط

9/2 = 4.5 → 5

3- نجد القيمة التي في الوسط

5 = 3

4- نبدل العنصر الذي في الوسط ونكتب المصفوفة

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 3 & 6 \\ 2 & 3 & 9 \end{bmatrix}$$

نأخذ الجزء التالي من المصفوفة وهو أيضا (3*3) ونرتبه تصاعديا

$$\begin{bmatrix} 3 & 5 & 4 \\ 15 & 6 & 7 \\ 3 & 9 & 11 \end{bmatrix}$$

1- نرتبه تصاعديا

3 3 4 4 **6** 7 9 11 15

2- نحدد العنصر في الوسط

9/2 = 4.5 → 5

3- نجد القيمة التي في الوسط

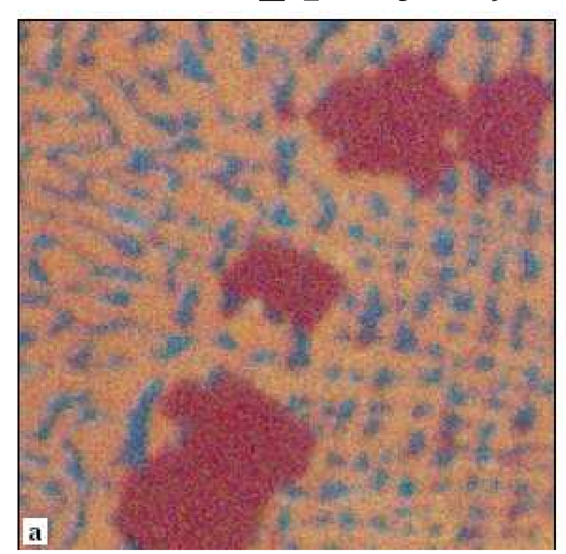
5 = 6

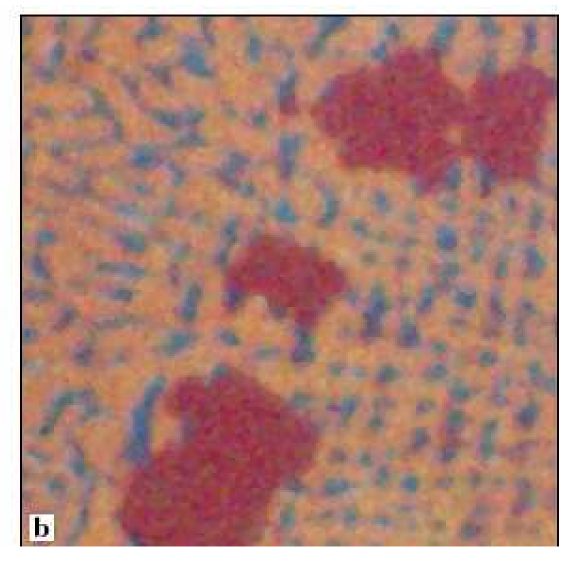
4- نبدل العنصر الذي في الوسط ونكتب المصفوفة

$$\begin{bmatrix} 3 & 5 & 4 \\ 15 & 6 & 7 \\ 3 & 9 & 11 \end{bmatrix}$$

وبالتالي تصبح المصفوفة الناتجة كتالي :

$$\begin{bmatrix} 2 & 3 & 5 & 4 \\ 1 & 3 & 6 & 7 \\ 2 & 3 & 9 & 11 \end{bmatrix}$$





شكل(15): مرشح الوسيط Median Filter

3- مرشح التحسين Enhancement Filter

التحسين يتكون من نوعين من المرشحات وهما :

1- Laplacian Enhancement

2- Difference Enhancement

يستخدم النوع الأول لتحسين تفاصيل الصورة في جميع الاتجاهات وله ماسكين هما :

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

أما النوع الثاني فيستخدم لتحسين تفاصيل الصورة باتجاه واحد وله أربع ماسكات

$$\begin{bmatrix} 0 & 0 & 0 \\ +1 & +1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

ماسك بالاتجاه الأفقي

$$\begin{bmatrix} 0 & +1 & 0 \\ 0 & +1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

ماسك بالاتجاه العمودي

$$\begin{bmatrix} 0 & 0 & +1 \\ 0 & +1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

ماسك بالاتجاه القطري الثانوي

$$\begin{bmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

ماسك بالاتجاه القطري الرئيسي

مثال // طبق ماسك Laplaci بالماسك الأول وماسك Difference القطري الرئيسي على جزء الصورة التالية ؟؟

$$\begin{bmatrix} 2 & 3 & 5 & 4 \\ 1 & 5 & 6 & 7 \\ 2 & 3 & 9 & 11 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & 5 \\ 1 & 5 & 6 \\ 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

-3 – 1 + 25 – 6 – 3 = 12

$$\begin{bmatrix} 3 & 5 & 4 \\ 5 & 6 & 7 \\ 3 & 9 & 11 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

-5 – 5 + 30 – 7 – 9 = 4

أما في حالة الDifference

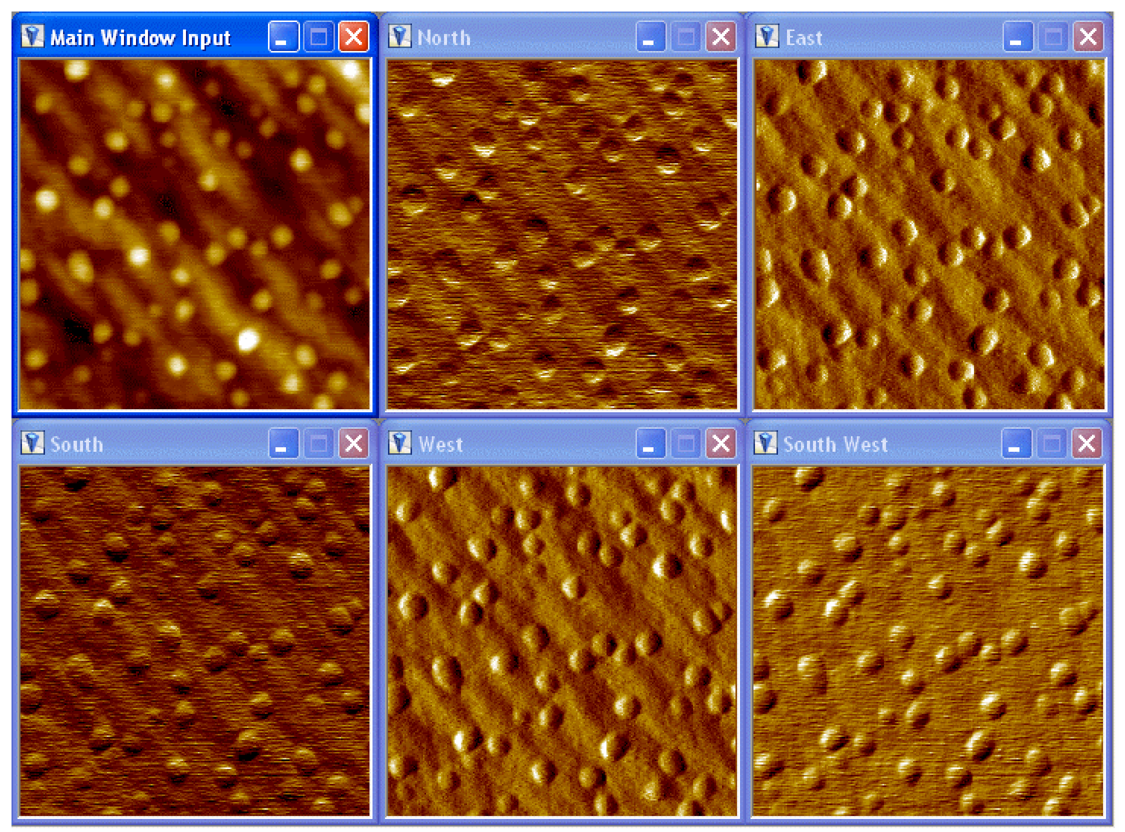
$$\begin{bmatrix} 2 & 3 & 5 \\ 1 & 5 & 6 \\ 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

2 + 5 – 9 = -2

$$\begin{bmatrix} 3 & 5 & 4 \\ 5 & 6 & 7 \\ 3 & 9 & 11 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

3 + 6 – 11 = -2

أذا في الرسم البياني نعتمد على الموقع وليس على القيمة أي أن ال12 موقعها (1 ، 1).



شكل(16):تحسين الصورة بجميع الاتجاهات Difference Enhancement

8-2 تقليص الصور (Image Quantization):

الفرق بين الضغط والتقليص : تقليص الصورة هي عملية نقل بيانات الصورة بإزالة بعض معلومات الصورة بإسقاط مجموعه عناصر الصورة إلى نقطة واحدة تتم عملية التقليص هذه (تكميم) .
أما الضغط : الصورة نفسها نتعامل معها كملف بينما التقليص قد يحذف جزء من الصورة ونتعامل مع القيم للصورة.

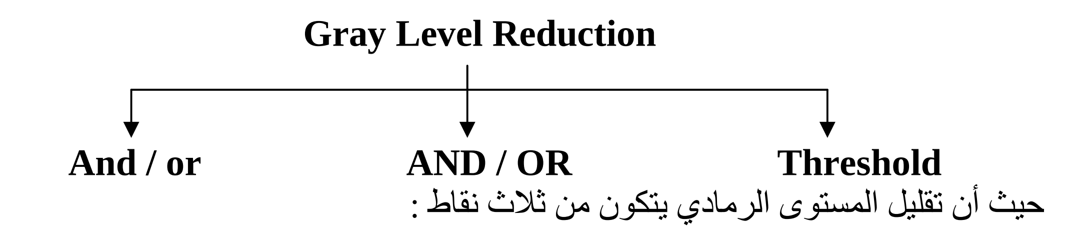
يوجد مجالان لتقليص الصورة هما :

1- Gray Level Reduction
أي نقوم بتقليل المستويات اللونية للصورة وهنا يتم على I(r, c) .

2- Special Reduction
وهنا يتم العمل على إحداثيات عناصر الصورة (r ,c) الموقع مثلا (1,1)

1- Gray Level Reduction

يمكن أن نوضح أقسامة كالأتي:



حيث أن تقليل المستوى الرمادي يتكون من ثلاث نقاط :

أ- الطريقة الأولى Threshold

يتم اختيار قيمة معينة من المستويات اللونية هذه القيمة تسمى عتبة فأي قيمة من بيانات الصورة أعلى من قيمة العتبة تصبح قيمتها واحد وإذا أقل تصبح قيمتها صفر ، يعني يتم هنا تحويل العدد ذات المستويات اللونية 256 إلى صور ثنائية .

مثال //

إذا كانت قيمة العتبة 127 طبقها للقيم التالية:

Gray Level
11
100
129
200
251

الحل // نرى أن أعلى قيمة هي 251 وأقل قيمة هي 11

حسب الصورة الثنائية العتبة هي 127 فتكون

Gray level	Threshold
11	0
100	0
129	1
200	1
251	1

مثال // إذا كانت قيمة العتبة 127 طبقها للقيم التالية:

الحل// هذه القيمة كلها تكون أقل من العتبة 127 فيكون أصفار ففي هذه الحالة نحدد الأكبر والأقل ونأخذ الوسط فيكون أقل قيمة 2 وأكبر قيمة 25 أذا الوسط بينهم هو 12 أو 13

Gray level	Threshold
7	0
11	0
2	0
4	0
20	1
25	1

ب- الطريقة الثانية : هي عملية الAND ، OR عدم استخدام الماسك
وهنا تتم تقليل عدد البتات لكل عنصر (Bit Per pixel)

مثال //
نريد أن نقلص أو نقلل المعلومات لثمان بتأت أذا كان الاحتمال القياسي لنا هو 256 للمستوى الرمادي إلى 32 مستوى استخدام طريقة الAND لتوضيح ذلك أقل رقم من كل خانة بالAND
الحل //
256 ÷ 8 = 32

0 123 4 5 6 7 **8** 9 10 11 12 13 14 15 **16**....................................... 256

0 8 16 …………………..256

إذا يجب تقليص هذه الأطياف الملونة من 256 إلى 32 طيف معناها كل 8 بتأت نضعها في خانة بعدها نأخذ أقل قيمة في كل خانة فيكون هنا الخانة الأولى أقل قيمة 0 والثانية 8 والثالثة 16 إلى أن تصل إلى 256 بحيث يكون عدد هذه الأرقام المستخرجة هي 32 رقم .

أما أذا كانت طريقة OR نأخذ أكبر رقم في الخانة لماذا ؟

الجواب // لأنة الOR لا تأخذ صفر والأرقام تكون أقل من الAND برقم واحد يبدأ من الخانة الثانية) فتكون ....32 15 7 الOR يأخذ أكبر رقم من كل خانة

مثال // أذا كان لديك عدد المستويات القياسية (256) المطلوب تقليلها إلى 16 ؟
الجواب // يعني يكون كل 16 بت بخانة واحدة فيكون التقسيم على 16 فنبدأ بالخانة الأولى من 0 إلى 16 وهكذا .. 16 = 16 ÷ 256

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 **16**....................................... 256

3- الطريقة الثالثة الAND والOR باستخدام الماسك

تستخدم هذه الطريقة لتقليص الصورة (تقليلها) باستخدام ماسك معين .

مثال //

إذا كان لديك الماسك التالي المطلوب استخدام طريقة الماسك للAND لتقليص هذا الجزء من الصورة اعتمادا على عدد البتات لكل عنصر وهي 8 ؟

$$\begin{bmatrix} 0 & 10 & 7 \\ 255 & 242 & 1 \\ 15 & 16 & 20 \end{bmatrix}$$

الحل //

قانون المستويات الرمادية $\mathbf{2^K}$

$$\mathbf{2^k = 8}$$
$$\mathbf{K = 3}$$

أي 8 تأتي من (0 ... 7)

وبالثنائي ال7 تصبح (111) فإذا الماسك هو عبارة عن 8 بتأت ويساوي00000111

ومن ثم نأخذ رقم رقم من المصفوفة ونحوله إلى الصيغة الثنائية ونضربه بالماسك

فنأخذ الرقم صفر يكون بالثنائي 00000000

وبعد ضربه بالمسك يكون

00000111
00000000
____
00000000

وال10 ونحوله إلى ثنائية هي 1010

00000111
00001010
____
00000010

أما رقم 255 فهو

11111111
00000111
____
00000111

وهكذا لبقية أرقام المصفوفة

$$\begin{bmatrix} 00000000 & 00001010 & 00000111 \\ 00000111 & & \\ & & \end{bmatrix} \begin{bmatrix} 0 & 2 & 7 \\ 7 & & \\ & & \end{bmatrix}$$

Special Reduction -2

التقليل ألحيزي يتم بثلاث طرق

1- المعدل   2- الوسيط   3- التنقيص

الطريقة الأولى : المعدل

وهي الطريقة التي تتم بأخذ مجموعة من العناصر المتجاورة واخذ المعدل لها .

مثال // المطلوب استخدام طريقة المعدل لجزء الصورة التالية

$$\begin{bmatrix} 9 & 8 \\ 11 & 22 \end{bmatrix}$$

الحل //

$$(9 + 8) / 2 = 8$$
$$(11 + 22) / 2 = 16$$
$$\begin{bmatrix} 8 \\ 16 \end{bmatrix}$$

أما إذا كان المعدل الكلي للصورة فيكون

Total Average of Image = (33 + 17) / 4 = 13

أما في حالة أذا كانت المعدل لصفوف فيكون بمجموع الصف على عدد الأرقام في الصف الواحد

الطريقة الثانية : الوسيط

في هذه الطريقة يتم ترتيب عناصر الصورة تصاعديا واخذ القيمة التي في الوسط .

مثال // لديك المصفوفة التالية المطلوب 1- أخذ الوسيط لكل العناصر2- باستخدام ماسك معين ؟

$$\begin{bmatrix} 9 & 8 & 7 & 3 \\ 4 & 9 & 5 & 2 \\ 1 & 3 & 4 & 6 \end{bmatrix}$$

الحل //

1- نرتب الأرقام تصاعديا فتصبح

1 2 3 3 4 **4** 5 6 7 8 9 9

أذا الوسيط هو العنصر السادس أي

$$12 \div 2 = 6$$

2- أما في حالة استخدام الماسك ونفرض الماسك هو 3*3 ونقوم بترتيب العناصر تصاعديا فيكون ترتيب عناصر المصفوفة الأولى وهي 3*3

$$\begin{bmatrix} 9 & 8 & 7 \\ 4 & 9 & 5 \\ 1 & 3 & 4 \end{bmatrix}$$

1 3 4 4 **5** 7 8 9 9

$$9 \div 2 = 4.5 \rightarrow 5$$

فإذا الوسيط هو العنصر الخامس وقيمته 5

فيكون ترتيب عناصر المصفوفة الثانية وهي 3*3

$$\begin{bmatrix} 8 & 7 & 3 \\ 9 & 5 & 2 \\ 3 & 4 & 6 \end{bmatrix}$$

2 3 3 4 **5** 6 7 8 9

$9 \div 2 = 4.5 \rightarrow 5$

فإذا الوسيط هو العنصر الخامس وقيمته أيضا 5

الطريقة الثالثة التنقيص

تتم بحذف بعض بيانات الصورة فمثلا تقليل حجم الصورة بمقدار 2 . فيتم هنا بأخذ كل صف أو عمود من الصورة وحذف الصف والعمود الذي يليه

مثال // لديك جزء الصورة التالي المطلوب التنقيص بمقدار 2 حسب الأعمدة ؟؟

$$\begin{bmatrix} 9 & 8 & 7 & 3 \\ 4 & 9 & 5 & 2 \\ 1 & 3 & 4 & 6 \end{bmatrix}$$

فيكون حذف العمود الثاني والرابع أي $2 - 1 = 1$

فتصبح المصفوفة كتالي

$$\begin{bmatrix} 9 & 7 \\ 4 & 5 \\ 1 & 4 \end{bmatrix}$$

أما إذا كان التنقيص بمقدار 3 فنقوم بحذف عمودين وهما الثاني والثالث أي $3 - 1 = 2$

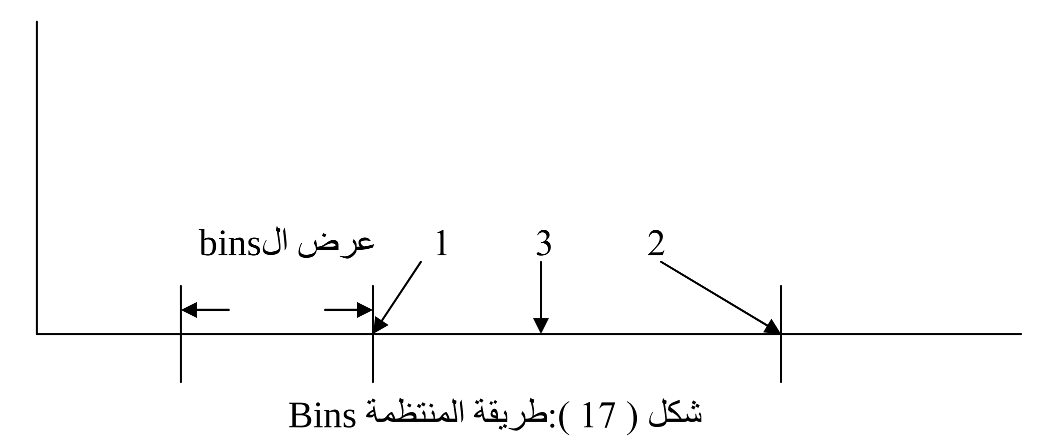
فتصبح المصفوفة كتالي

$$\begin{bmatrix} 9 & 3 \\ 4 & 2 \\ 1 & 6 \end{bmatrix}$$

وفي هذه الحالة أذا كان المطلوب التنقيص بمقدار 3 صفوف فيكون الجواب لا يمكن لأن المصفوفة مكونة من 3 صفوف وليس 4 .

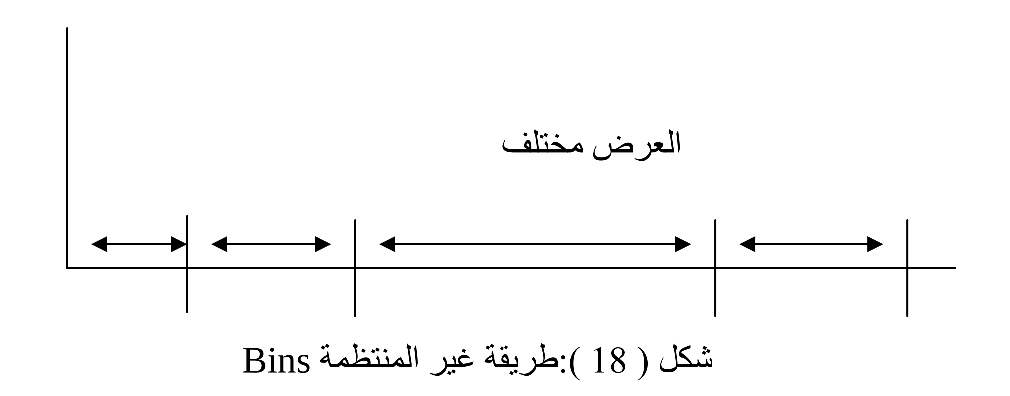
ملاحظة // على الرغم من أن طريقة الANDوالOR هي طريقة فعالة جدا لكنها ليست مرنة أذا كانت أحجام المصفوفة (الصورة) كبيرة ، لذلك يجب استخدام طريقة أخرى هي طريقة الBins وأن سبب وجود طريقة الBins هو لأن الطريقة السابقة هي فعالة ولكنها ليست مرنة أما طريقة الBins هي ليست سريعة ولكنها منتظمة ويمكن أن تحاكي النظرة الشخصية للنظام حسب لوغاريتمات معينة للبنات والشكل العام لها هو :

1)Low end
2) High end
3) Middle



شكل (17):طريقة المنتظمة Bins

Bins : هي وحدات صغيرة بتجميعها نكون نظام متكامل لتقليص الصورة والمسافة بين بن وأخر هي متساوية ،أما الشكل الثاني لها فهي أن عرض البنات مختلف :



شكل (18):طريقة غير المنتظمة Bins

مثال // لديك المصفوفة التالية المطلوب تطبيقها حسب البنات

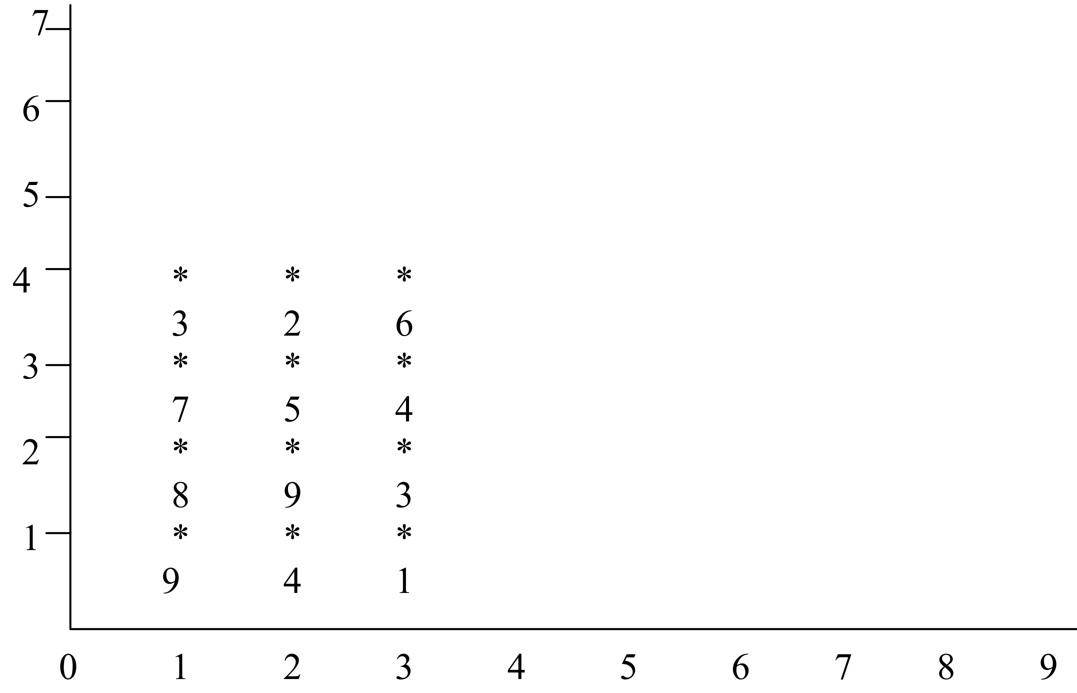
$$\begin{bmatrix} 9 & 8 & 7 & 3 \\ 4 & 9 & 5 & 2 \\ 1 & 3 & 4 & 6 \end{bmatrix}$$

1- في حالة أذا كان العرض ثابت بين البنات .

2- في حالة أذا كان العرض مختلف .

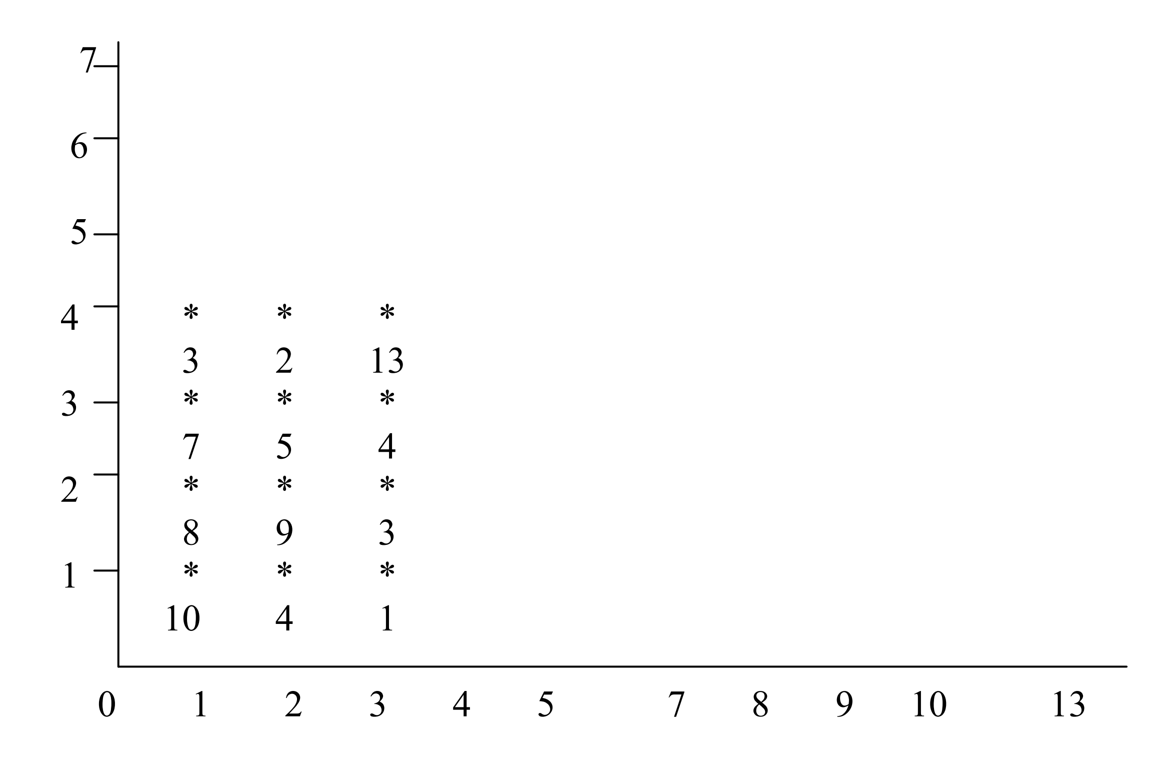
الحل //

نلاحظ هنا أن أقل قيمة هي 1 وأعلى قيمة هي 9 فعند رسم الbin نرسم كوحدات واحدة يعني واحد لكل Bin يعني مثلا العنصر واحد بالمصفوفة هو بالموقع (1 ، 3)



أما في حالة عدم تساوي المسافات بين الـbins فنأخذ المصفوفة التالية كمثال توضيحي

10	8	7	3
4	9	5	2
1	3	4	13



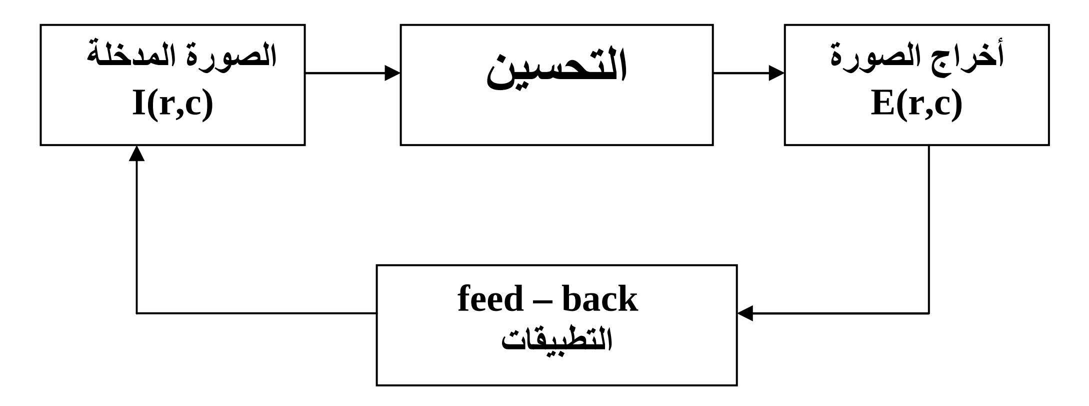
الفصل الثالث

تحسين الصورة الرقمية

1-3 تحسين الصورة (Image Enhancement Techniques):

تقنيات تحسين الصورة : هي عبارة عن التقنيات التي تستخدم لتحديد التشدد أو الحدود الموجودة في الصورة لإبراز مميزات الصورة وخصائصها وتحليلها .

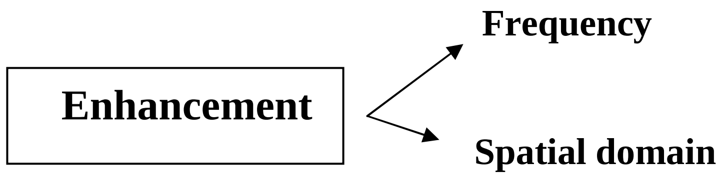
تستخدم هذه التقنيات في تطبيقات متعددة باستخدام ما يسمى بالتغذية الرجعية من المدخلات إلى المخرجات لإعطاء موديلات التحسين المختبرة بطبيعتها بعد التطوير .



شكل(19): معالجة تحسين الصورةThe image enhancement process

أحدى تطبيقات الFeed Back أو طرائق لتحسين الصورة طريقة التعديل والتحسين مستخدم المستوى الرمادي كذلك تسمى بتعديل المستوى الرمادي من خلال العمليات التي تطبق على النقطة باستخدام الدوال وتغير هذه النقاط مستخدمين معادلة تسمى بالMapping equation معادلة المطابقة وهي معادلات خطية وغير خطية نوعا ما وتعدل بواسطة موديلات خطية وتطابق مع النموذج الأصلي للمستوى الرمادي أي قيمته مع القيم المخصصة الأخرى .

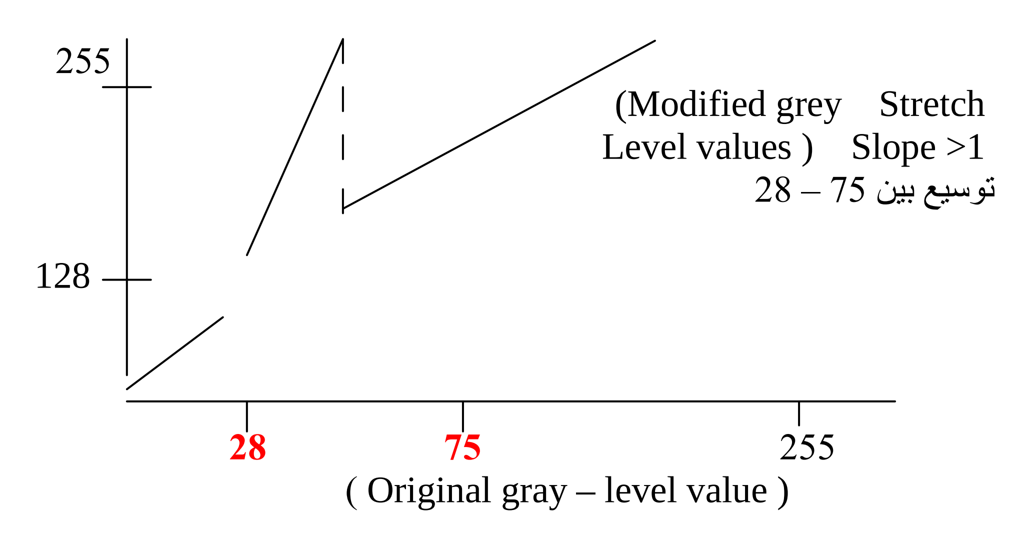
تتضمن هذه التطبيقات ما يسمى بتحسين التباين وما يسمى بتحسين المميزات (الخواص).



معتمدين على العمليات الأولية التي تطبق على المستويات الرمادية أما أن تكون العمليتين السابقتين (التباين – الخواص) أما أن تكون تقليص أو أن تكون تكبير للمستويات الرمادية وهنا أبسط الأنواع أن تأخذ المعادلات الخطية ونطبق هذه العملية بحيث نقوم بضغط المستوى الرمادي أو ضغط المستوى الرمادي أذا كان المدى أكبر من الواحد فنعتبره توسيع .

1- عندما نأخذ صورة اعتمادا على التباين (هو بالألوان)
2- أما أذا اعتمدنا على الخواص فهو (أي الألوان سوف تبرز أكثر)

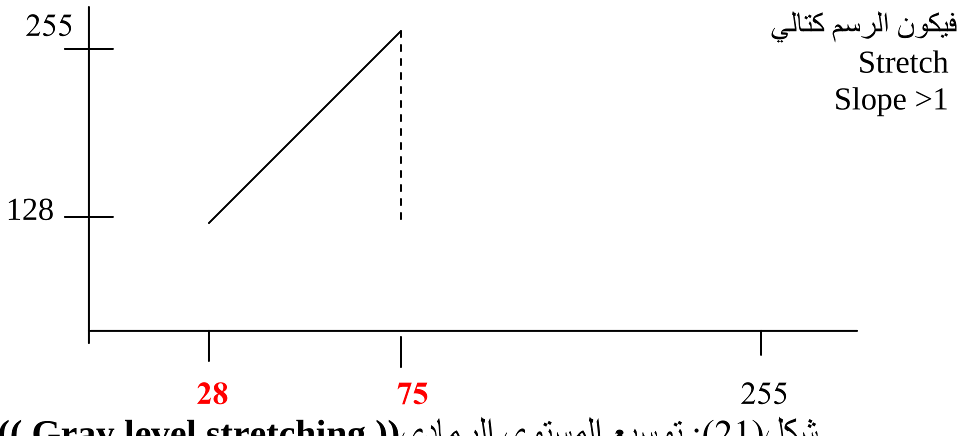
مثال //
فسر الرسم التالي معتمدين على المدى Slop ؟



شكل(20):توسيع المستوى الرمادي((Gray level stretching))

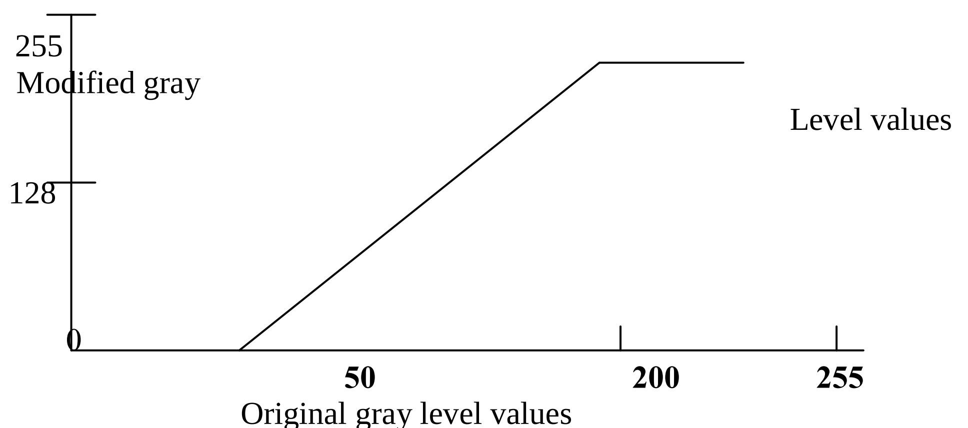
الحل //
1- المعالجة تبدأ من الصفر إلى 255 معتمدا على المستويات الرمادية .
2- بما أن المدى الSlop هو أكبر من الواحد فهو توسيع وإذا كان المدى من 0 إلى الواحد فهو تقليص ، وإذا لم يعطى الSlop فنحن نعرف المدى لأن الرسم يبدأ من بعد الواحد يعني يبدأ من 28 .
3- النقاط التي تم توسيعها هي من 28 إلى 75
4- الرسم بدون قصقصة Chipping يعني توجد نقاط بين 28 وال255

فيكون الرسم كتالي



شكل(21): توسيع المستوى الرمادي((Gray level stretching))

مثال // لديك الرسم التالي فسر ذلك ؟



شكل (22): توسيع المستوى الرمادي بقصقصة النهايات

Gray level stretching with clipping at ends

الحل //

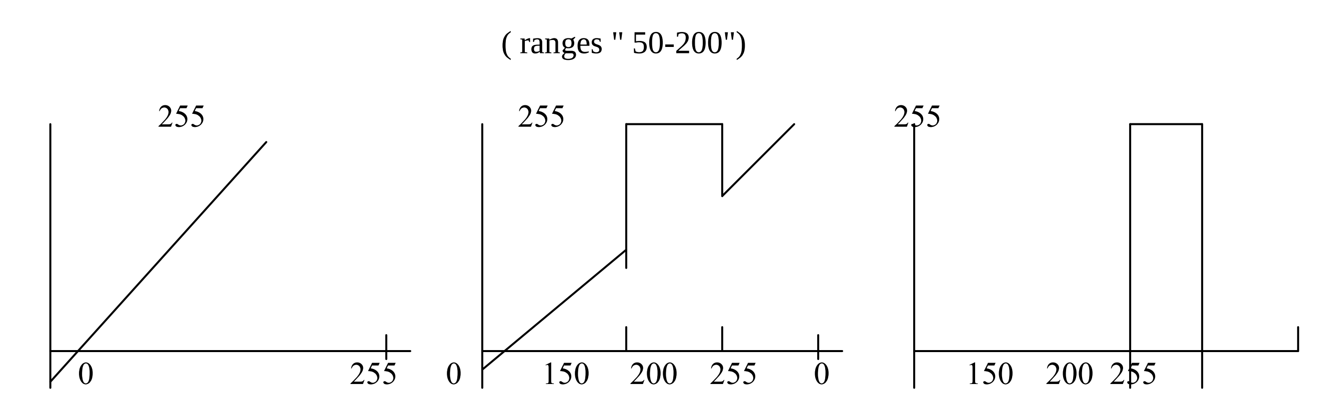
1- لأنه يبدأ من الصفر إلى 255 فهو يعتمد على المستويات الرمادية .
2- المدى الSlop هو أكبر من الواحد لأنه بدأ من 50 وهي عملية توسيع .
3- النقاط التي تم توسيعها هي بين 50 إلى 255
4- الرسم بقصقصة Chipping لا توجد نقاط بين 50 وال255

يمكن تقسيم معادلات المطابقة هي

1- معادلات المطابقة لتوسيع أو تقليص بقصقصة
2- معادلات المطابقة لتوسيع أو تقليص بدون قصقصة
3- معادلات المطابقة تستخدم لاستخلاص الخواص يطلق عليها معادلة شريحة مستوى الشدة Intensity Level Slicing

هنا يتم اختيار قيم مستوى رمادي مخصصة للشدة وتطبيقها بحيث تعطي إضاءة عالية والمثال التالي يوضحها

(ranges " 50-200")



The original gray levels **The desired gray level Range (change the other value)** **The desired gray level range (change the other value)**

شكل(23): انواع القصقصة

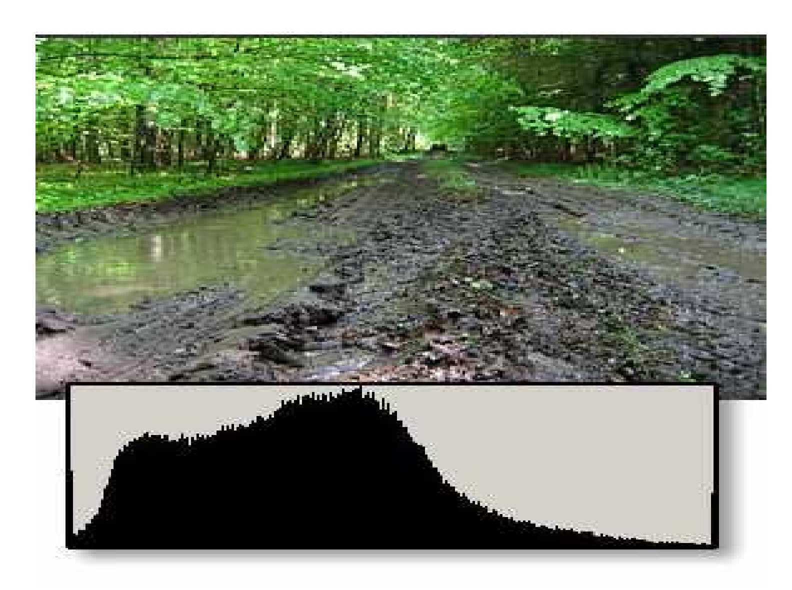
الرسم الأول : هو مستويات رمادية أصلية لم تحدد هل هي بقصقصة أو توسيع أو تقليص
الرسم الثاني : هو عبارة عن المجال المخصص هنا الذي يبتدئ من 150 وينتهي ب200 والمجال عام أي مجرد تغير بالقيم
الرسم الثالث : حصلنا على مستويات رمادية متخصصة متغيرة القيم .
وهذه الحالات الثلاثة هي دالة المطابقة لشريحة مستوى الشدة يعني مستوى الشدة هنا على تقليص أو توسيع أو قصقصة .

2-3 تعديل المخطط Histogram modification:

يقوم المخطط الذي يستخدم المستويات الرمادية للصورة بتوزيع هذه المستويات الخاصة بالصورة بحيث يجعل جزء من الصورة الذي يحتوي على المعلومات يملأ أو يكون المخطط وبقية المساحة تكون فارغة اعتمادا على القيم الخاصة بنقاط الصورة الخاصة للمخطط .

توجد العديد من هذه المستويات المعدلة يمكن أن نذكرها كالأتي :

1- المدرج التكراري مع انتشار صغير للمستويات المتباينة (الرمادية) Low Contrast Image.
2- المدرج التكراري مع انتشار كبير للمستويات الرمادية المتباينة High Contrast Image
3- المدرج التكراري المتجمع عند النهاية الواطئة Dark Slide Image
4- المدرج التكراري المتجمع عند النهاية العليا White Slide Image

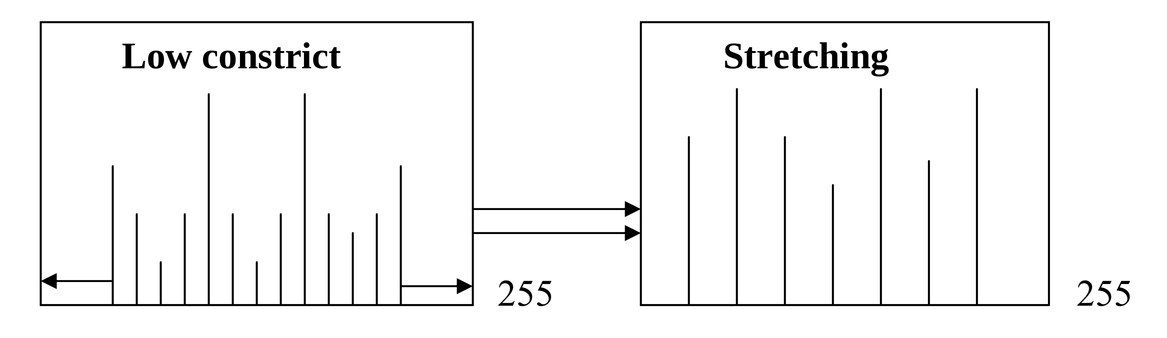


شكل(24):صورة مع المدرج التكراري لها

وتتم عملية تغير المدرج التكراري بثلاث طرق : Histogram Scaling

1- توسيع المدرج التكراري Histogram Stretching
2- تقليص المدرج التكراري (Histogram Shrink (compressed
3- الشريحة الخاصة بالمخطط Slide of Histogram

*** الطريقة الأولى : توسيع المدرج التكراري Histogram Stretching**



شكل(25): **توسيع المدرج التكراري**

يمكن توسيع المدرج التكراري حسب القانون التالي :

$$\text{Stretch}(I(r,c)) = \left[\frac{I(r,c) - I(r,c)_{min}}{I(r,c)_{max} - I(r,c)_{min}}\right][\text{Max} - \text{Min}] + \text{Min} \quad ....(6)$$

حيث أن :

1- قيمة المستوى الرمادي الكبرى في الصورة $I(r,c)_{max}$
2- قيمة المستوى الرمادي الصغرى في الصورة $I(r,c)_{min}$
3- تعتمد قيم المستوى الرمادي الصغرى والكبرى المحتملة أي Max & Min (0 ، 255)

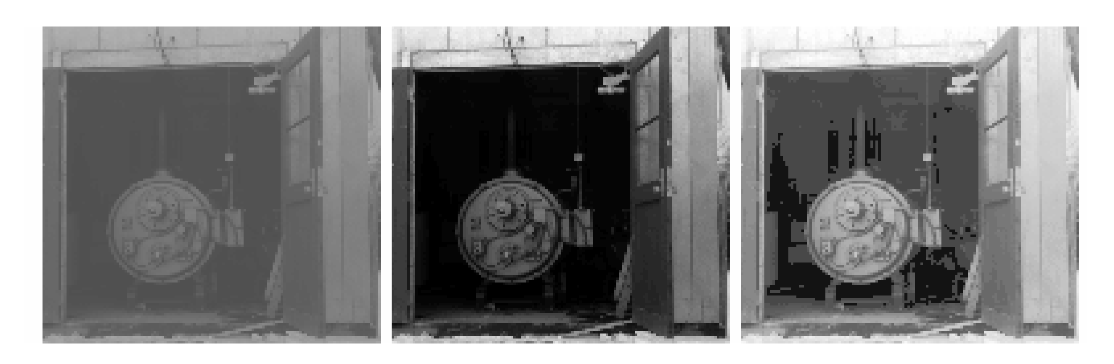
مثال // لديك جزء الصورة التالي المطلوب توسيع هذا الجزء من الصورة باستخدام طريقة توسيع المدرج التكراري ؟

$$\mathbf{I}_{\mathbf{Sub}} = \begin{bmatrix} 7 & 12 & 8 \\ 20 & 9 & 6 \\ 10 & 15 & 1 \end{bmatrix}$$

$$\text{stretch}(I(r,c)) = \left[\frac{I(r,c) - I(r,c)_{min}}{I(r,c)_{max} - I(r,c)_{min}}\right][\text{Max} - \text{Min}] + \text{Min}$$

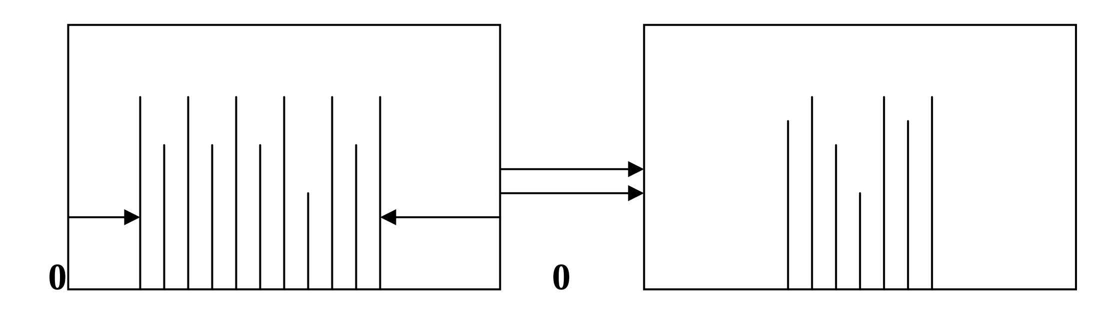
= [7-1 / 20-1] * [255 – 0] + 0 = 76.5
= [12-1 / 20-1] * [255 – 0] + 0 = 127.5
= [8-1 / 20-1] * [255 – 0] + 0 = 76.5
= [20-1 / 20-1] * [255 – 0] + 0 = 255
= [9-1 / 20-1] * [255 – 0] = 7.6
= [6-1 / 20-1] * [255 – 0] + 0 = 51
= [10-1 / 20-1] * [255 – 0] + 0 = 102
= [15-1 / 20-1] * [255 – 0] + 0 = 178.5
= [1-1 / 20-1] * [255 – 0] + 0 = 0

$$I_{Stretch} = \begin{bmatrix} 76.5 & 127.5 & 76.5 \\ 255 & 7.6 & 51 \\ 102 & 178.5 & 0 \end{bmatrix}$$



شكل(26): توسيع التباين لصورةContrast stretching

*** الطريقة الثانية تقليص المدرج التكراري histogram shrinking**



شكل(27): **الطريقة الثانية تقليص المدرج التكراري**

يمكن تقليص المدرج التكراري حسب القانون التالي :

$$Shrink(I(r,c)) = \left[\frac{Shrink_{Max} - I(r,c)_{Min}}{I(r,c)_{Max} - I(r,c)_{Min}}\right][I(r,c) - I(r,c)_{Min}] + shrink_{Min}$$

……(7)

حيث أن :-

1- قيمة المستوى الرمادي الكبرى في الصورة هي $I(r,c)_{max}$
2- قيمة المستوى الرمادي الصغرى في الصورة هي $I(r,c)_{min}$
3- تعتمد على قيم المستوى الرمادي $Shrink_{min}$ & $Shrink_{max}$ الكبرى والصغرى المحتملة أي (0 ، 255) .

مثال // لديك جزء الصورة التالي المطلوب تقليص هذا الجزء من الصورة باستخدام طريقة تقليص المدرج التكراري ؟

$$\mathbf{I}_{Sub} = \begin{bmatrix} 7 & 12 & 8 \\ 20 & 9 & 6 \\ 10 & 15 & 1 \end{bmatrix}$$

$$Shrink(I(r,c)) = \left[\frac{Shrink_{Max} - I(r,c)_{Min}}{I(r,c)_{Max} - I(r,c)_{Min}}\right] [I(r,c) - I(r,c)_{Min}] + shrink_{Min}$$

= [255-1 / 20-1] * [7 – 1] + 0 = 79.8
= [255-1 / 20-1] * [12 – 1] + 0 = 146.3
= [255 -1 / 20-1] * [8 – 1] + 0 = 93.1
= [255-1 / 20-1] * [20 – 1] + 0 = 252.7
= [255-1 / 20-1] * [9 – 1] + 0 = 106.4
= [255-1 / 20-1] * [6 – 1] + 0 = 66.5
= [255-1 / 20-1] * [10 – 1] + 0 = 119.7
= [255-1 / 20-1] * [15 – 1] + 0 = 186.2
= [255-1 / 20-1] * [1 – 1] + 0 = 0

$$I_{Shrink} = \begin{bmatrix} 79.8 & 146.3 & 93.1 \\ 252.7 & 106.4 & 66.5 \\ 119.7 & 186.2 & 0 \end{bmatrix}$$

*** الطريقة الثالثة: أزاحه المدرج التكراري Histogram slide**

يمكن أزاحه المدرج التكراري لمسافة معينة حسب القانون التالي :

Slide (I (r,c)) = I (r,c) + OFFSET…………..(8)

حيث أن :-

Offset : هي الكمية التي يزاح بها المدرج التكراري لمسافة ما .

مثال // لديك جزء من الصورة التالي المطلوب أزاحه هذا الجزء لمسافة مقدارها 10 وحدات باستخدام طريقة Histogram slide

$$I_{Sub} = \begin{bmatrix} 12 & 7 & 8 \\ 20 & 9 & 6 \\ 10 & 15 & 1 \end{bmatrix}$$

Slide (I (r,c)) = I (r,c) + OFFSET
= 7 + 10 = 17
= 12 + 10 = 22
= 8 + 10 = 18
= 20 + 10 = 30
= 9 + 10 = 19
= 6 + 10 = 16
= 10 + 10 = 20
= 15 + 10 = 25
= 1 + 10 = 11

$$I_{Slide} = \begin{bmatrix} 17 & 22 & 18 \\ 30 & 19 & 16 \\ 20 & 25 & 11 \end{bmatrix}$$



شكل (28): أزاحه المدرج التكراري لصورة

3-3 المدرج التكراري المخصص : Histogram specification

هو معالجة الصورة بواسطة استخدام المدرج التكراري وتعديل هذا المدرج بحيث يكون مطابق للمدرج المخصص .

الطريقة :

1- أيجاد جدول التخطيط (Mapping Table) ويسمى (H) ويعني المدرج التكراري المنظم .
2- نقوم بوصف مدرج تكراري مخصص .
3- نقوم بإيجاد جدول التخطيط للمدرج التكراري المنظم من خلال وصف قيم المدرج التكراري .
4- نقوم بإيجاد جدول يعتمد على القيم الأصلية للخطوات من 1 إلى خطوة 3 .

مثال // لديك الجدول التالي المطلوب أيجاد المدرج التكراري المخصص ؟

Gray level	**Number of pixel (histogram values)**
0	**10**
1	**8**
2	**9**
3	**2**
4	**14**
5	**1**
6	**5**
7	**2**

Step1 : 10 , 18 , 27 , 29 , 43 , 44 , 49 , 51

Step2 : 10 + 8 + 9 + 2 + 14 + 1 + 5 + 2 = 51
10/51 , 18/51 , 27/51 , 29/51 , 43/51 , 44/51 , 49/51 , 51/51 .

Step3 : Multiply these values by Maximum gray level values in this case 7 and round the result to the integer . After this done we option 1,2,4,4,6,6,7,7 .

Step4 : Map the original values to the results from step3 by one to one correspondence .

Original gray level values	Histogram equalized values
0	**1**
1	**2**
2	**4**
3	**4**
4	**6**
5	**6**
6	**7**
7	**7**

Round ((10/51) * 7) = 1
Round ((18/51) * 7) = 2
Round ((22/51) * 7) = 4
|
|
|
Round ((51/51) * 7) = 7

Example :-

Step1 :

Original Gray level value-O	Number of pixel in histogram-H
0	**1**
1	**2**
2	**4**
3	**4**
4	**6**
5	**6**
6	**7**
7	**7**

Step2 : specify the histogram

Gray level value	Number of pixels in histogram
0	**1**
1	**5**
2	**10**
3	**15**
4	**20**
5	**0**
6	**0**
7	**0**

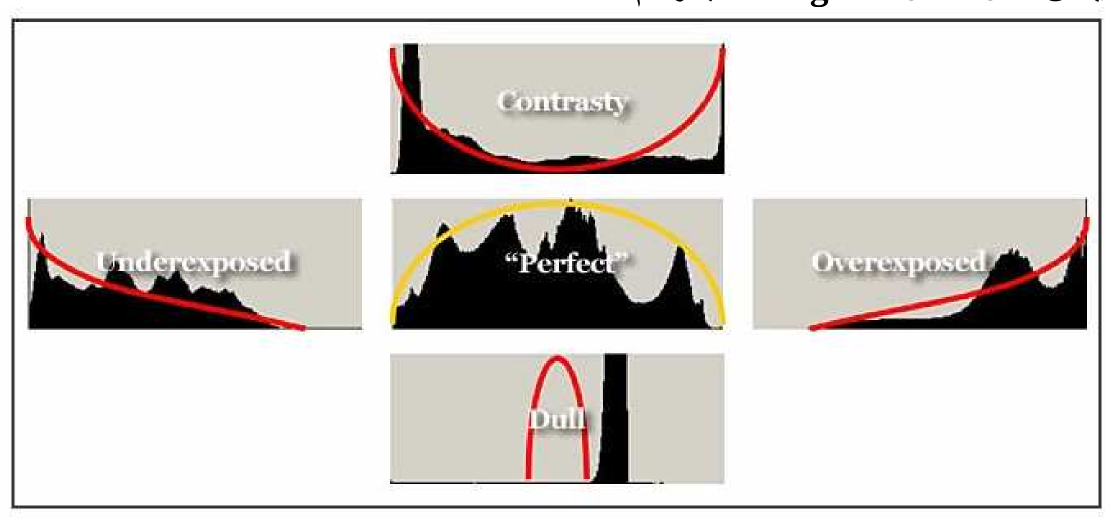
Step3 :

Gray level value	Histogram equalized – S
0	Round (1/51) * 7 = 0
1	Round (6/51) * 7 = 1
2	Round (16/51) *7 = 2
3	Round (31/51) * 7 = 4
4	Round (51/51) * 7 = 7
5	Round (51/51) * 7 = 7
6	Round (51/51) * 7 = 7
7	Round (51/51) * 7 = 7

Step4:

O	H	S	M
0	1	0	1
1	2	1	2
2	4	2	3
3	4	4	3
4	6	7	4
5	6	7	4
6	7	7	4
7	7	7	4

يمكن تمثل أشكال Histogram بالرسم أدناه



شكل(29):أشكال Histogram للصور الرقمية

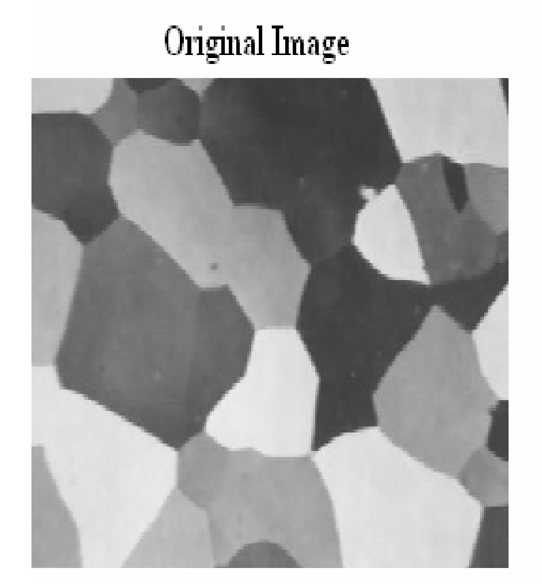
4-3 كشف الحواف للصورة Edge / Line Detection For Image:

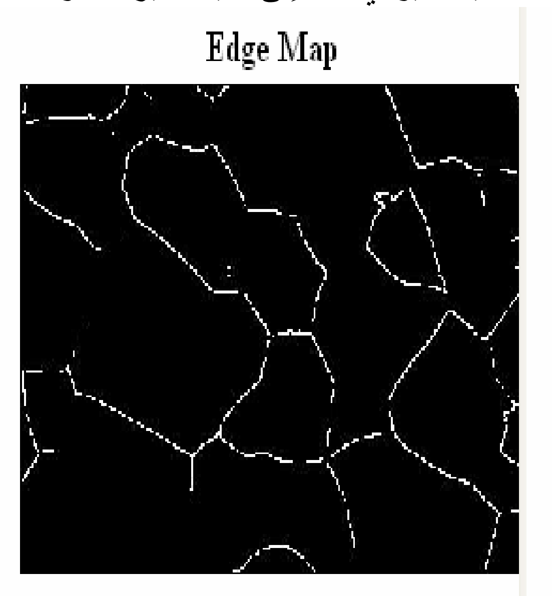
شدة إضاءة الحواف تكون أعلى من مجاوراتها حيث تحدد من خلال الفرق بين الكسل وبين المتجاورات له يكون كبير جدا أو يتم اكتشاف الحواف باستخدام ما يسمى بالتلفيف (Convolution Mask) حيث أن الحافة هي البحث عن أكبر تغيير بالدالة (دالة الإضاءة) بعض عمليات كشف الحواف تحمل ما يسمى باتجاه الحافة (Edge Direction) متجه الحافة و(Edge Magnitude) قيم الحافة .

يوجد معاملين عند تطبيق عمليات كشف الحواف :

1- حجم الماسك المستخدم في تحديد الحافة :
فإذا كان كبير سيكون أقل حساسية لكشف الحواف مثلا مصفوفة 3*3 أكثر حساسية من 7*7

2- قيمة العتبة (Gray Level Threshold) :
فإذا كانت قليلة سيؤدي ذلك إلى تقليل تأثير الضوضاء .





شكل(30): صورة أصلية مع تحديد الحواف لها

ملاحظة //
تعتمد جميع عمليات كشف الحواف على أن معلومات الحافة يمكن الحصول عليها من العلاقة بين عناصر الصورة ومجاوراتها فإذا كان عنصر الصورة يشبه مجاوراته لا توجد حافة ..
أما أذا كان لا يشبهه وهناك فرق كبير بينهم عندئذ توجد حافة ..

3-5 طرائق إيجاد الحافة واتجاهها
Edge / Line Detection and direction For Image

توجد العديد من الطرائق الخاصة لإيجاد الحافة واتجاه الحافة يمكن أدراجها كالأتي:

1- Roberts Operation

يستخدم هذا الكاشف لكشف نقاط الحافة بدون أن نعطي أي معلومات عن اتجاه الحافة أي فقط (Edge Magnitude) ويأخذ هذا الكاشف شكلين :

I (r-1 , c-1)	**I (r-1 , c)**
I (r , c-1)	**I (r , c)**

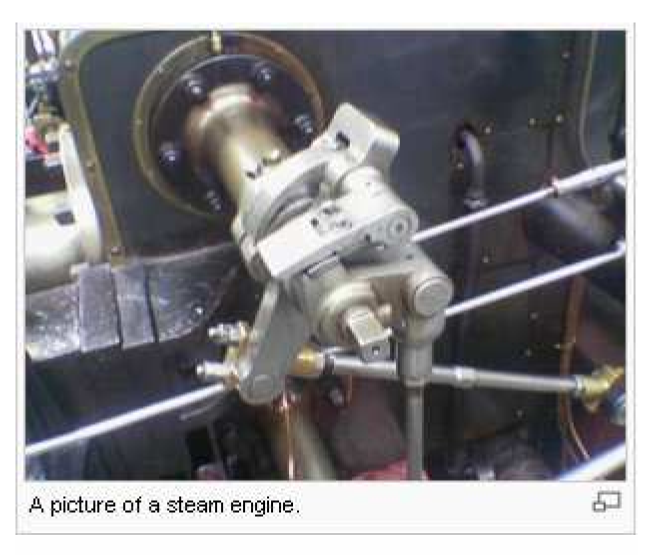
أ - يتم حساب الحافة بأخذ الجذر ألتربيعي لمجموع مربعات فروق الأقطار كما في المعادلة :

$$Edge = \sqrt{[I (r ,c) - I (r-1, c-1)]^2 + [I (r,c-1) - (r-1 ,c)]^2} \quad \textbf{...(9)}$$

ب – الطريقة الثانية تكون باستخدام المعادلة :

$$Edge = \left| [I (r ,c) - I (r-1 , c-1)] + [I (r , c-1) - I (r-1 , c)] \right| \quad .....(10)$$

يتم إيجاد مجموع فروقات الأقطار المتجاورة باستخدام القيمة المطلقة





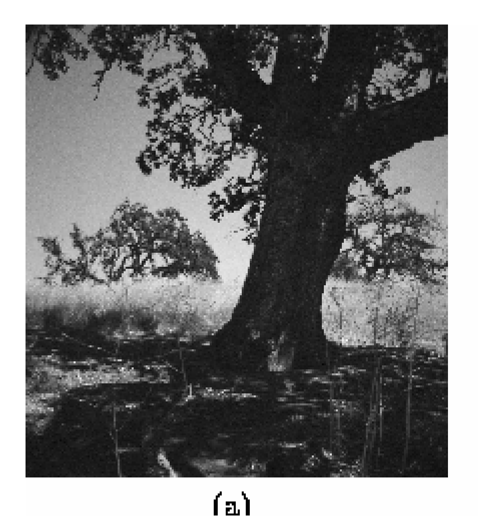
شكل(31):ايجاد الحافة باستخدام Roberts Operation

ملاحظة //
يستخدم الشكل الثاني B في التطبيق العملي حيث تكون العملية أسهل وأسرع يدويا أما الطريقة A فيفضل استخدامها بالحاسوب .

2- SOBEL Operation

يستخدم هذا الكاشف لإيجاد قيمة الحافة واتجاه الحافة بالاتجاهين الأفقي والعمودي حيث يستخدم هنا ماسك هو :

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{row mask} \rightarrow S1 \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{column mask} \rightarrow S2$$


(a)


(b)

شكل(32): أ- صورة اصلية ب- تاثير **SOBEL Operation** عليها

ويتم تحديد قيمة الحافة عن طريق القانون التالي :-

$$\textbf{Edge Magnitude} = \sqrt{S_1^2 + S_2^2} \quad ....(11)$$

S_1 : هي row mask

S_2 : هي column mask

ملاحظة //

نطبق مصفوفة الS_1 للصفوف على أول 3*3 من الصورة ثم نطبق مصفوفة الS_2 للأعمدة على 3*3 من الصورة نفسها ثم نجمع القيمتين الناتجتين ثم نضيف مجموعهما مع المربعات إلى مركز أول 3*3 أي في منتصف المركز .

مثال // طبق قانون SOBEL للمصفوفة التالية ؟

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & 4 & 3 \\ 1 & 8 & 6 \end{bmatrix}$$

الحل // نضرب 3*3 من الصورة بالS_1

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & 4 & 3 \\ 1 & 8 & 6 \end{bmatrix} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = -5$$

$$= -3 - 18 - 7 + 1 + 16 + 6 = -5$$

نفس الجزء نضربه بالS_2

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & 4 & 3 \\ 1 & 8 & 6 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = -5$$

$$= -3 + 7 - 4 + 6 - 1 + 6 = 11$$

$$= \sqrt{5^2 + 11^2} = \sqrt{25 + 121} = 7$$

فنضع هذه القيمة في منتصف المصفوفة أي ال7 بدل ال4 فتصبح المصفوفة

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & \color{red}{7} & 3 \\ 1 & 8 & 6 \end{bmatrix}$$

وبعدها نأخذ الجزء الثاني بتزحيف المصفوفة إلى التالية مع أخذ المصفوفة الجديدة بنظر الاعتبار عند التزحيف أي يستخدم الرقم الجديد الذي وضع في وسط الماسك .

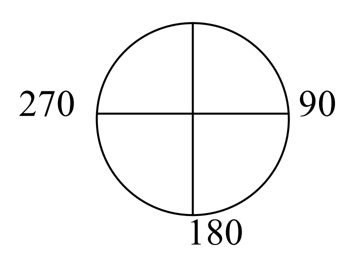
$$\text{Edge Direction} = \text{TAN}^{-1} [-5 / 11] = 0.045$$

اتجاه الحافة يستخدم القانون التالي :

$$\textbf{Edge Direction} = \textbf{TAN}^{-1} \left[\frac{S_1}{S_2} \right] \quad \textbf{..........(12)}$$

إذا كان السؤال الاتجاه لكل ماسك يكون بأخذ الاتجاه له بتطبيق القانون Edge direction على كل ماسك ...

بينما إذا كان الاتجاه الكلي للصورة نقوم بجمع الماسكات ... + $S_1 + S_2 + S_3$ وبعدها نأخذ لها قانون الاتجاه Edge direction مع رسم الاتجاه له ...



واجب // طبق قانون الSOBEL على المصفوفة التالية لإيجاد الحافة وإيجاد قيمة اتجاه الحافة والاتجاه الكلي ؟

$$\begin{bmatrix} 3 & 27 & 3 & 9 \\ 4 & 7 & 16 & 8 \\ 8 & 3 & 6 & 5 \end{bmatrix}$$

Prewitt Operator -3

يشبه الماسك السابق الSOBEL حيث يتم تعريفة كتالي :-

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

(Row mask P1) **(Column mask P2)**

ولإيجاد قيمة الحافة Edge Magnitude

$$\textbf{Edge Magnitude} = \sqrt{P_1^2 + P_2^2} \qquad \ldots.(13)$$

أما لإيجاد الاتجاه للحافة فيكون : -

$$\textbf{Edge Direction} = \textbf{TAN}^{-1} \left| \frac{P_1}{P_2} \right| \qquad \ldots\ldots\ldots\ldots\ldots(1$$

مثال // طبق قانون الPrewitt على المصفوفة التالية لإيجاد الحافة وإيجاد قيمة اتجاه الحافة والاتجاه الكلي ؟

$$\begin{bmatrix} 3 & 9 & 7 & 1 \\ 2 & 4 & 3 & 3 \\ 1 & 8 & 6 & 4 \end{bmatrix}$$

الحل // نأخذ الجزء الأول 3*3

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & 4 & 3 \\ 1 & 8 & 6 \end{bmatrix} * P_1 \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = -4$$

$$= -3 - 9 - 7 + 1 + 8 + 6 = -4$$

$$\begin{bmatrix} 3 & 9 & 7 \\ 2 & 4 & 3 \\ 1 & 8 & 6 \end{bmatrix} * P_2 \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = 10$$

$$= -3 + 7 - 2 + 3 - 1 + 6 = 10$$

$$\text{Edge Magnitude} = \sqrt{-4^2 + 10^2} = \sqrt{16 + 100} = \sqrt{116} = 10.7 = 11$$

نضع هذه القيمة في منتصف المصفوفة أي بدل ال4

$$\text{Edge Direction} = TAN^{-1} \left| -4 / 10 \right| = TAN^{-1} (0.4) = 0.36 = 0$$

أذا الاتجاه هو الشمال

أما الجزء الثاني من المصفوفة فيكون كتالي

$$\begin{bmatrix} 9 & 7 & 1 \\ 11 & 3 & 3 \\ 8 & 6 & 4 \end{bmatrix} * P_1 \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = 1$$

$$= -9 - 7 - 1 + 8 + 6 + 4 = 1$$

$$\begin{bmatrix} 9 & 7 & 1 \\ 11 & 3 & 3 \\ 8 & 6 & 4 \end{bmatrix} * P_2 \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = -20$$

$$= -9 + 1 - 11 + 3 - 8 + 4 = -20$$

$$\text{Edge Magnitude} = \sqrt{1^2 + -20^2} = \sqrt{1 + 400} = \sqrt{401} = 20.2 = 20$$

نضع هذه القيمة في منتصف المصفوفة أي بدل ال3 فتصبح المصفوفة

$$\begin{bmatrix} 3 & 9 & 7 & 1 \\ 2 & 11 & 20 & 3 \\ 1 & 8 & 6 & 4 \end{bmatrix}$$

$$\text{Edge Direction} = TAN^{-1} \left| 1 / -20 \right| = TAN^{-1} (0.05) = 0.04 = 0$$

أذا اتجاه المصفوفة شمال

أذا الاتجاه الكلي لمصفوفة الصورة يكون كالتالي :

$$P_1 = P_1 + P_1 = -4 + 1 = -3$$

$$P_2 = P_2 + P_2 = 10 - 20 = -10$$

$$\text{Edge Magnitude} = \sqrt{-3 + (-10)} = \sqrt{9 + 100} = \sqrt{109} = 10.4 = 10$$

$$\text{Edge Direction} = TAN^{-1} \left| -3 / -10 \right| = TAN^{-1} (0.29) = 0$$

أذا اتجاه المصفوفة شمال



شكل(33):صورة اصلية وتاثير **Prewitt Operator**

4- Kirsch Compass Masks

هو أحد المعاملات الخاصة بتحديد الحواف واتجاه الحافة سمي بهذا الاسم لأنه يحدد المحيط أو الإطار الخاص بالصورة ويقوم بأخذ ماسك مفرد في كل مره وتدويره بالاتجاهات المختلفة . وتوجد ثمان ماسكات له وهن :-

$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & 3 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & 3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

K0 **K1** **K2** **K3**

$$\begin{bmatrix} 5 & 3 & 3 \\ 5 & 0 & 3 \\ 5 & 3 & 3 \end{bmatrix} \quad \begin{bmatrix} 3 & 3 & 3 \\ 5 & 0 & 3 \\ 5 & 5 & 3 \end{bmatrix} \quad \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ 5 & 5 & 5 \end{bmatrix} \quad \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 5 \\ 3 & 5 & 5 \end{bmatrix}$$

K4 **K5** **K6** **K7**

وللحصول على قيمة الحافة يكون عبارة عن قيمة الماسك في تلفيف الماسك

Magnitude of Edge = Max Value of Mask Convolution

ويتم استخراجها من الحصول على أعلى قيمة من تلفيف الماسك

Direction of Edge = Is The Direction of Maximum Value

يتم استخدام كل الماسكات على الصورة فيكون أعلى جزء هو اتجاه الحافة .

كل ماسك من هذه الماسكات الثمانية نعمل لها تلفيف مع الماسك الخاص بالصورة (3*3) في كل مره أعلى قيمة هي التي نأخذها أما مجموع قيم المصفوفة الكلية تتم بجمع قيم الماسكات للجزء الواحد .

مثال // لديك الصورة التالية المطلوب تطبيق معامل Kirsch على هذه الصورة للجزأين الأولين للصورة فقط ؟

7	3	0	1	2	1
2	9	3	3	2	1
1	4	5	1	1	1

F_1 F_2

نضرب ماسك $F_1 * K_0$ واستخراج النتيجة ونضعها في S_0 وكذلك نضرب ماسك $F_1 * K_1$ واستخراج النتيجة ونضعها في S_1 وهكذا لحد ما نضرب $F_1 * K_7$ ونضع النتيجة في S_7 وقيمة الحافة تكون أعلى قيمة واتجاهها نفس اتجاه أعلى قيمة أما مجموع قيم الماسكات لهذا الجزء نقوم بجمع جميع قيم الـS

الحل //

$$K_0 * F_1 = S_0$$
$$K_1 * F_1 = S_1$$
$$K_2 * F_1 = S_2$$
$$\vdots$$
$$K_7 * F_1 = S_7$$

أما بالنسبة للجزء الثاني

$$K_0 * F_2 = S_0$$
$$K_1 * F_2 = S_1$$
$$K_2 * F_2 = S_2$$
$$\vdots$$
$$K_7 * F_2 = S_7$$

توجد مشكله هنا في ماسكات الـKirsch

1- القيم الموجودة في الماسكات هي قيم كبيرة وغير مرتبه
2- وجود حسابات كثيرة يمكن الخطاء فيها
3- قيم الصفر التي تبين الاتجاه في الماسكات تكون قليلة

5- Robinson Compass Masks

في هذا المعامل أيضا توجد ثمان ماسكات وهن :-

$$R_0 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad R_1 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad R_2 = \begin{bmatrix} 1 & 2 & 2 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad R_3 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$R_4 \qquad\qquad R_5 \qquad\qquad R_6 \qquad\qquad R_7$$

قيمة الحافة هي عبارة عن أعلى قيمة للحافات .
اتجاه الحافة هو أعلى قيمة للاتجاه المأخوذ .
وكذلك سهولة في الحاسوب وسهولة في إيجاد اتجاه الحافة



شكل(34): إيجاد الحافة واتجاهها باستخدام **Robinson**

LAPLACIAN OPERATORS -6

أخذنا هذا المعامل في تحسين الصورة ويتكون من ثلاث ماسكات مجموع القيم بالماسكات يجب أن ينتج قيمة تساوي واحد وذلك لكي نزيد من إضاءة الصورة أما معامل الLAPLACAN الموجود هنا فأنة يتكون من ثلاث ماسكات أيضا لكن مجموع معاملات الماسكات يجب أن يكون صفر .

$$\underset{L_2}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}} \quad \underset{L_1}{\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}} \quad \underset{L_0}{\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}}$$

LAPLACIAN MASKS

لماذا يفضل أن تكون مجموع قيم المعاملات = صفر ؟
وذلك لأن عملية المسح للبكسلات يجب أن تعمم مثل البكسلات الصورة بحيث نضع الخلفية سوداء وحدود اللون أبيض مثل الأشعة .

مثال // لديك الصورة التالية المطلوب تطبيق معامل LAPLACIAN على هذه الصورة لجميع ماسكاته ؟

$$\begin{bmatrix} 4 & 3 & 2 & 1 & 2 & 1 \\ 1 & 1 & 5 & 1 & 2 & 1 \\ 6 & 7 & 1 & 2 & 2 & 1 \end{bmatrix}$$

F_1 F_2

الحل //

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * \begin{bmatrix} 4 & 3 & 2 \\ 1 & 1 & 5 \\ 6 & 7 & 1 \end{bmatrix} = -12$$

$$L_0 * F_1 = -3 - 1 + 4 - 5 - 7 = -12$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} * \begin{bmatrix} 4 & 3 & 2 \\ 1 & 1 & 5 \\ 6 & 7 & 1 \end{bmatrix} = -15$$

$$L_1 * F_1 = 4 - 6 + 2 - 2 + 4 - 10 + 6 - 14 + 1 = -15$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} 4 & 3 & 2 \\ 1 & 1 & 5 \\ 6 & 7 & 1 \end{bmatrix} = -20$$

$$L_2 * F_1 = -4 - 3 - 2 - 1 + 8 - 5 - 6 - 7 - 1 = -20$$

لا يمكن تحديد اتجاه الماسك لأنه هو يقوم بتحديد الماسك داخل إطار .

أذا أراد كل الماسكات نطبق كل الماسكات ونختار الأعلى أما أذا أراد ماسك واحد فنختار أعلى قيمة من الماسك .

أذا أعطيت جزئيين من الصورة واختيار واحده من الماسكات

$L_2 * F_1 = -11$

$L_2 * F_2 = 5$

جزأين للصورة وثلاث ماسكات

أذا أراد معامل ماسك واحد مثلا L_2 نضرب هذا المعامل في الجزء الأول ونأخذ أكبر قيمة . أما كل الماسكات فنجزئ وأضرب الأول في الجزء الأول والثاني في الجزء الأول وهكذا أو نختار أعلى قيمة .

$L_0 * F_1 = -3$

$L_1 * F_1 = \mathbf{9}$

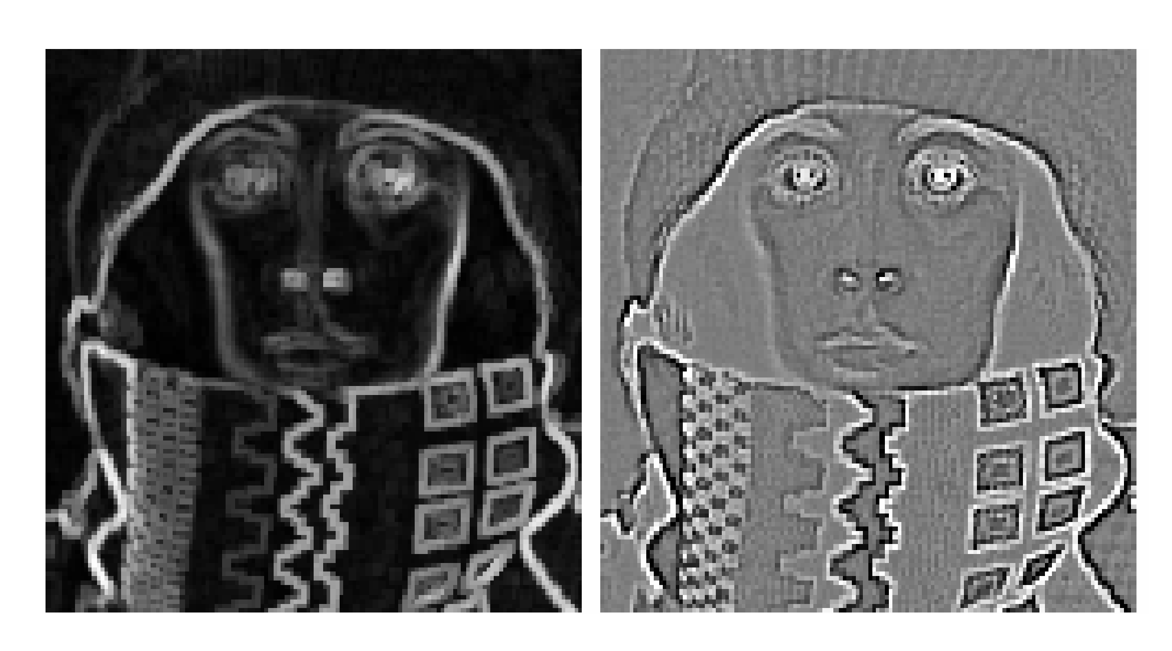
$L_2 * F_1 = -1$

$L_0 * F_2 = 2$

$L_1 * F_2 = \mathbf{13}$

$$L_2 * F_2 = 5$$

بما أنه نفس الماسك وهو أما تكون قيمته في كل مرة فيكون نفس اللون .



شكل(35): تأثير **LAPLACIAN MASKS** على الصورة

FREI – CHEN MASKS -7

هذا الماسك يعتبر فريد من نوعه في اكتشاف الحافات من حيث فكرته ، حيث يستخدم هنا معاملات للماسكات كمجموع للأوزان الخاصة بكل جزء في الصورة ومن ناحية أخرى عدد الماسكات هنا 9 ، والفكرة العامة له هو أنه الأربع ماسكات الأولى يأخذها Edge Sub Space والأربع ماسكات الثانية تستخدم لخطوط المساحة الجزئية Line Sub Space والماسك الأخير فهو يستخدم للمعدل Average Sub Space .

$$\frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & \sqrt{2} & -1 \end{bmatrix} \qquad \frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$$

F1 F2

$$\frac{1}{2\sqrt{2}}\begin{bmatrix} 0 & 1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix} \qquad \frac{1}{2\sqrt{2}}\begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}$$

F3 F4

$$\frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

F5 F6

$$\frac{1}{6}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \frac{1}{6}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

F7 F8

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

F9



شكل(36) تأثير : **FREI – CHEN MASKS** على الصورة

من بين الاختلافات هنا هو

1- ازدياد عدد الماسكات
2- توجد معاملات للماسكات

مثال // لديك الصورة التالية (جزء من صورة) المطلوب تطبيق ماسكات FREI – CHEN عليها

$$I_s = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

1

الحل // 1- نقوم بتطبيق كل الماسكات على هذا الجزء
- 1

$\frac{1}{2\sqrt{2}}$ [1(1) + 0 ($\sqrt{2}$) + 1 (1) + 1 (0) + 0 (0) + 1(-1) + 0(-$\sqrt{2}$)+ 1(-1)] = 0.

F1 → 0 F5 → -1
F2 → 0 F6 → 0
F3 → 0 F7 → 0
F4 → 0 F8 → -1

2 - نقوم باختيار القيم غير الصفرية
-3 نقوم بالعملية التالية وهي بضرب النتيجة المعطاة للماسك * الماسك مع معاملة + النتيجة المعطاة للماسك غير الصفري الثاني * الماسك مع معاملة + وهكذا

$$(-1)\left(\tfrac{1}{2}\right)\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} + (-1)\left(\tfrac{1}{6}\right)\begin{bmatrix}-2 & 1 & -2\\ 1 & 4 & 1\\ -2 & 1 & -2\end{bmatrix}$$

$$+ (2)\left(\tfrac{1}{3}\right)\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix} = Is$$

أذا النتيجة تظهر نفسها أي نفس جزء الصورة أذا توجد حافة وإذا لم تنتج نفس النتيجة معناها لا توجد حافة والخلفية مستمرة .

واجب // لديك الصورة التالية (جزء من صورة) المطلوب تطبيق ماسكات FREI – CHEN عليها ؟؟

$$I_s = \begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}$$

ملاحظة //

لكي نستخدم ماسكات الFREI – CHEN حتى تدل على اتجاه الحافة لابد من أن نستخدم القانونين التاليين :-

$$\mathbf{Cos\ \Theta = \sqrt{\frac{M}{S}}}\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(15)$$

$$\mathbf{1\text{-}\quad M = \sum_{k\in\{e\}} (I_s , f_k)^2}\quad \ldots\ldots\ldots\ldots\ldots\ldots\mathbf{(16)}$$

2- $S = \sum_{k=1}^{\infty} (I_s , f_k)^2$ ....(17)

حيث أن :-

1- قيمة الـM هي أحدى الماسكات غير الصفرية ويتم اختيارها بعد ضرب معاملها مع الصورة .
2- أما الـS فهي قيمة الماسكات المأخوذة الكلية نجمعها مع بعضها ومن ثم نربع الناتج .

ولتوضيح هذه المسألة نأخذ المثال السابق حيث توجد نتيجة ضرب الماسكات مع جزء الصورة غير الصفرية نطبق عليها قانون اتجاه الحافة

$F_5 = -1$
$F_8 = -1$
$F_9 = 2$

وبما أن الـM هي أحدى الماسكات غير الصفرية فنأخذ الأقرب حيث أن :-

$M = F_5 = -1$
$M = (F_5)^2 = (-1)^2 = 1$

أما الـS فهي مجموع كل نتائج الماسكات الصفرية وغير الصفرية مع تربيعها

$S = 0^2 + 0^2 + 0^2 + 0^2 + (-1)^2 + 0^2 + 0^2 + (-1)^2 + (2)^2 = 6$

ثم نقوم بإيجاد اتجاه الحافة

$\sqrt{\text{Cos } \Theta} = \sqrt{M / s} = 1 / 6$

بعدها نرسم الزاوية التي تمثل ذلك

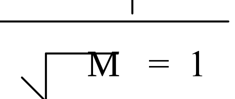
الوتر

$\sqrt{S} = \sqrt{6}$

$\sqrt{(6)^2} = \sqrt{n^2 + 1^2}$

$n^2 = 6 - 1$

$\sqrt{n} = 5 = 2.236$



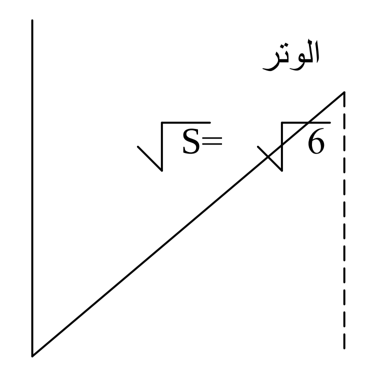
أما في حاله لو أخذنا الـM هي F_9

$M = F_9 = 2$
$M = (F_9)^2 = (2)^2 = 4$

ثم نقوم بإيجاد اتجاه الحافة

$\sqrt{\text{Cos } \Theta} = \sqrt{M / s} = 4 / 6$

بعدها نرسم الزاوية التي تمثل ذلك



$\sqrt{(6)^2} = (n)^2 + (4)^2$

$n^2 = 6 - 4$

$\sqrt{n} = \sqrt{2} = 1.414$

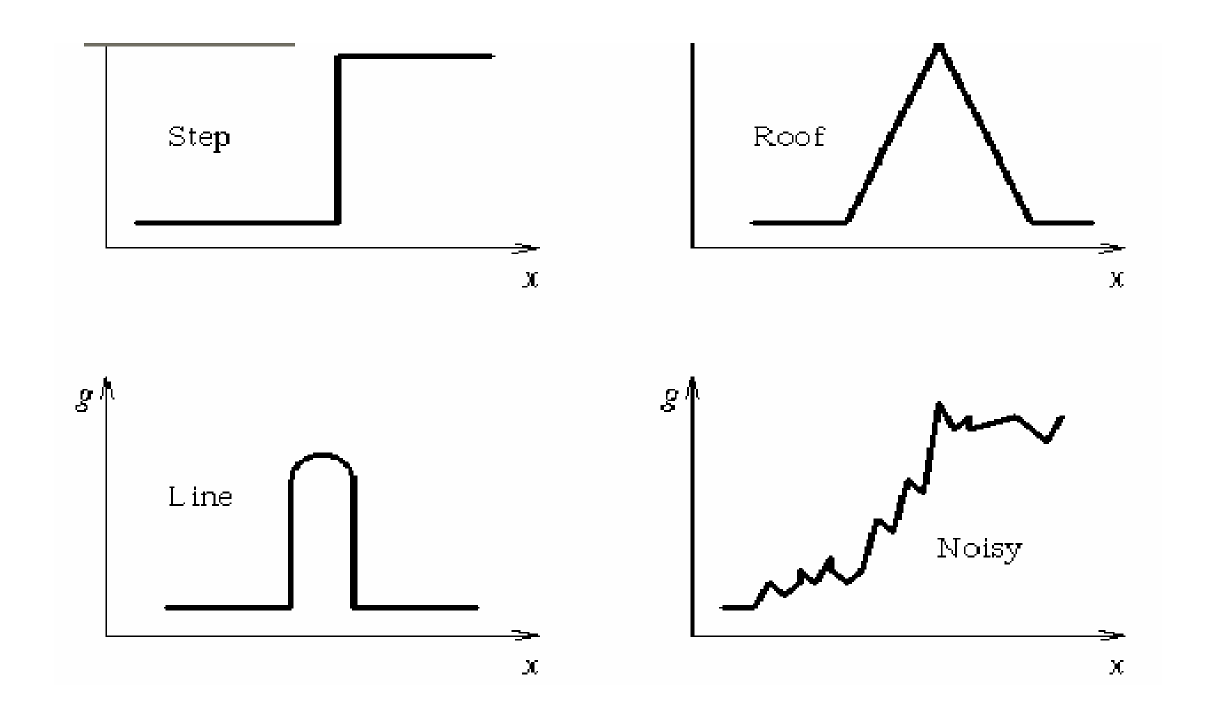
$M = \sqrt{4}$

شكل(37):زاوية اتجاه الحافة باستخدام فلتر FREI – CHEN

ملاحظة// يمكن أن نقسم أنواع الحافات إلى الأنواع التالية

1- خطوة
2- roof
3- خطي
4- مبعثر ضوضاء

والشكل التالي يوضح أشكال أنواع الحافاتroof



شكل(38): أنواع الحافات

الفصل الرابع

تنعيم وحدة الصورة الرقمية

1-4 زيادة حدة التفاصيل الصورة (IMAGE SHARPEING):

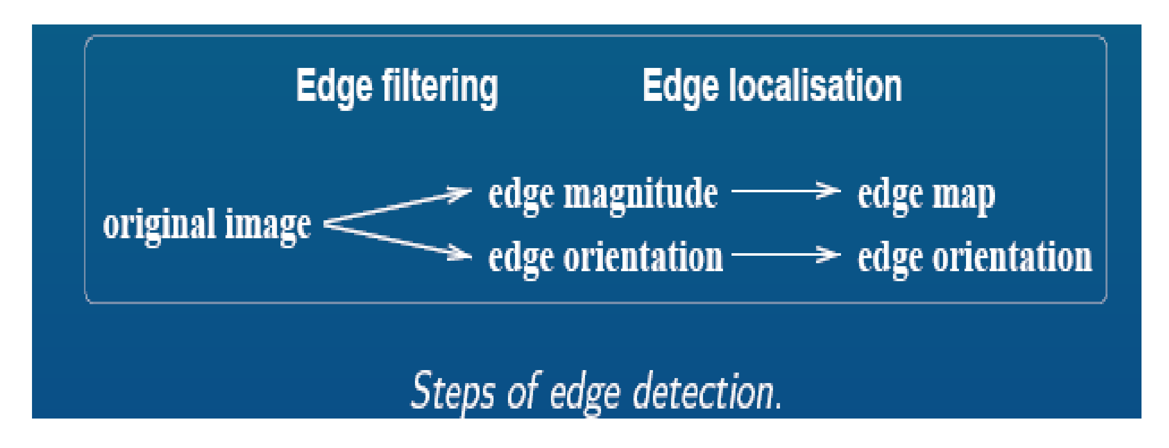
توجد طرق عديدة لإظهار حدة التفاصيل الخاصة بالصورة الرقمية يمكن إدراجها كالأتي:

أ- HIGH PASS FILTERING

زيادة حدة التفاصيل باستخدام مرشحات الإمرار العالي :

يمكن تطبيق عمليات زيادة حده التفاصيل بعد تطبيق أحد التحويلات (DCT) على عناصر الصورة حيث من المعروف أن الحواف التغيرات الحادة تكون مصاحبة لمادة عناصر التردد العالي .

يمكن الحصول على زيادة حده التفاصيل في المجال الترددي بعملية ترشيح التحرير العالي التي تحتفظ بعناصر التردد العالي وتهمل عناصر التردد الواطئ .



شكل(39):يمثل خطوات إيجاد الحافة

ب- HIGH – FREQUENCY EMPHASIS

زيادة حدة التفاصيل بالتأكيد على الترددات العالية :

يمكن استخدام تقنيات التأكيد على الترددات العالية لزيادة حدة التفاصيل في الصورة مع الاحتفاظ ببعض معلومات الترددات الواطئة وذلك بإضافة أزاحه تسمى Offset Value إلى دالة الترشيح يمكن الحصول على نفس النتائج في المجال ألحيزي باستخدام مرشحات يكون على شكل Mask يأخذ الهيئة التالية :-

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & x & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

حيث أن المتغير X يحدد كمية المعلومات في الترددات الواطئة التي يجب أن تبقى في الصورة الناتجة فإذا كانت قيمة X تساوي 8 سوف يكون المرشح من نوع High Pass Filter (الصورة الناتجة تحتوي على الحواف فقط) .

بينما أذا كانت قيمة X أكبر فسوف يتم الاحتفاظ بمعلومات أكثر من الصورة الأصلية وإذا كانت قيمة X أقل من 8 فسوف تكون الصورة الناتجة سالبة أو عكس الصورة الأصلية .

أذا كان حجم الMask أكبر سيؤدي ذلك إلى التأكيد أكثر للحواف .

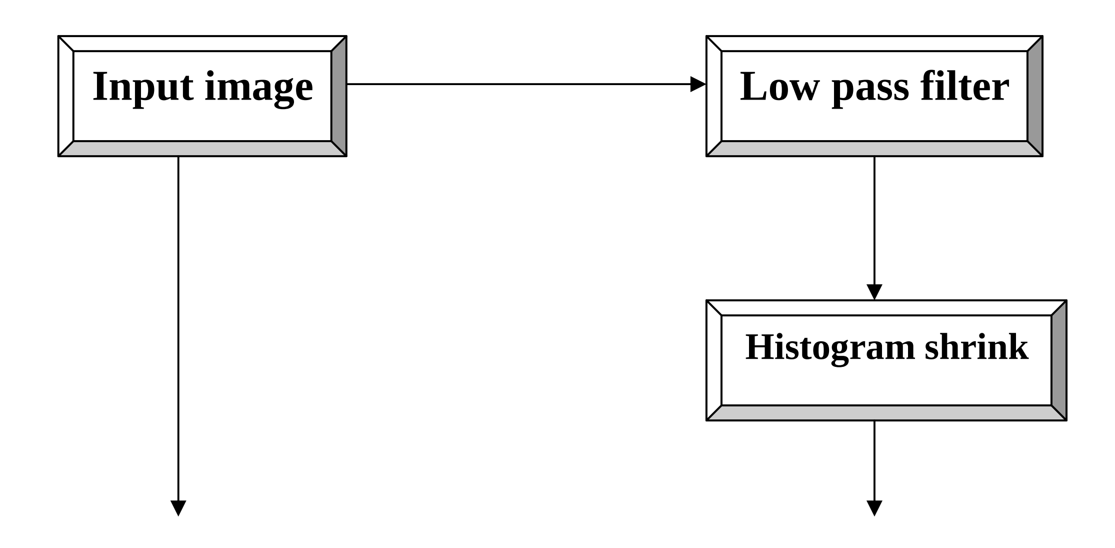
فمثلا // أذا كان حجم الMask 5*5

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & X & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

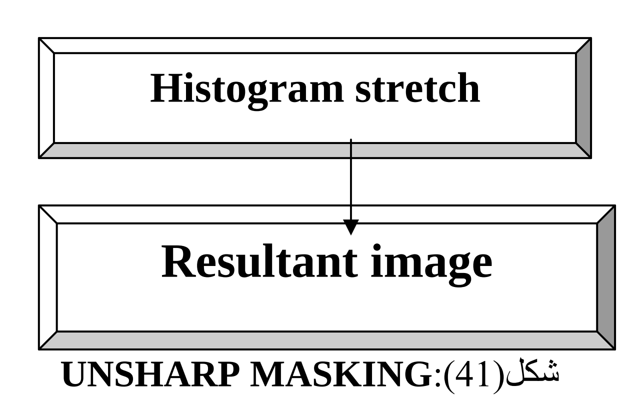
ج- UNSHARP MASKING

وهي خوارزميات لتحسين الصورة وذلك بزيادة حدة التفاصيل حيث يتم في هذه الخوارزمية جمع العديد من العمليات مثل الترشيح وعمليات تغير المدرج التكراري .

والمخطط الانسيابي لهذه الخوارزمية هو :-



Subtract image



شكل(41):**UNSHARP MASKING**

2-4 تنعيم الصورة **Image Smoothing:**

عمليات تنعيم الصورة تستعمل لها هدفين أساسيين هو

1- لجعل الصورة أكثر نعومة .
2- لتخلص من الNoise

تستعمل عمليات تنعيم الصورة لأضعاف الآثار الزائفة التي يمكن أن توجد في الصورة الرقمية وتتم عمليات التنعيم في المجالين ألحيزي والترددي حيث تتم في المجال ألحيزي بأخذ كل عنصر صورة والعناصر المجاورة له وإبعاد أي قيمة مختلفة عن هذه المجموعة وتتم عادة باستخدام مرشحات (Mean ، Median) .

أما في المجال الترددي فتتم باستخدام مرشحات الأحرار المنخفض Low Pass Filter بعد تطبيق أحد التحويلات على الصورة .

ويمكن كتابة خوارزمية تدوير ماسك التنعيم للصورة الرقمية كالأتي:

Algorithm 4.1: Rotated mask smoothing

1. Consider each image pixel (i, j).
2. Calculate dispersion in the mask for all possible mask rotations about pixel (i, j) according to equation (4.32).
3. Choose the mask with minimum dispersion.
4. Assign to the pixel $g(i, j)$ in the output image the average brightness in the chosen mask.

1- الطريقة ألحيزيه تقسم إلى :-

ا - مرشح المعدل Mean Filter (توسيط الجوار)

أن توسيط الجوار هي أحدى تقنيات المجال ألحيزي لتنعيم الصورة فإذا كان لدينا الصورة F(x,y) وأبعادها n*n فإن تطبيق التوسيط هو توليد صورة منعمة g (x, y) يتم الحصول عليها عند كل نقطة (x ,y) بأخذ متوسط القيم لعناصر الصورة المجاورة للعنصر (x ,y)

$$g (x ,y) = 1 / M \sum f (x, y) .... (18)$$

حيث M = عدد عناصر الMask أو الجوار

كلما كان حجم الMask سوف يحصل تنعيم أكثر للصورة .

ب- Median Filter (المرشح الأوسطي)

تستخدم المرشحات الأوسطية لتنعيم الصورة وذلك باستبدال قيمة المركز الMask المستخدم بالقيمة الوسطية .

مثال // أذا كان لدينا Mask بحجم 3*3

$$\begin{bmatrix} 100 & 25 & 20 \\ 20 & \mathbf{15} & 20 \\ 20 & 20 & 20 \end{bmatrix} \Longrightarrow \begin{bmatrix} 100 & 25 & 20 \\ 20 & \mathbf{20} & 20 \\ 20 & 20 & 20 \end{bmatrix}$$

15 20 20 20 **20** 20 20 25 100

Of Pixel = 9 / 2 = 4.5 = 5

2- الطريقة الترددية تتكون من :-

مرشحات الأحرار المنخفض Low Pass Filter

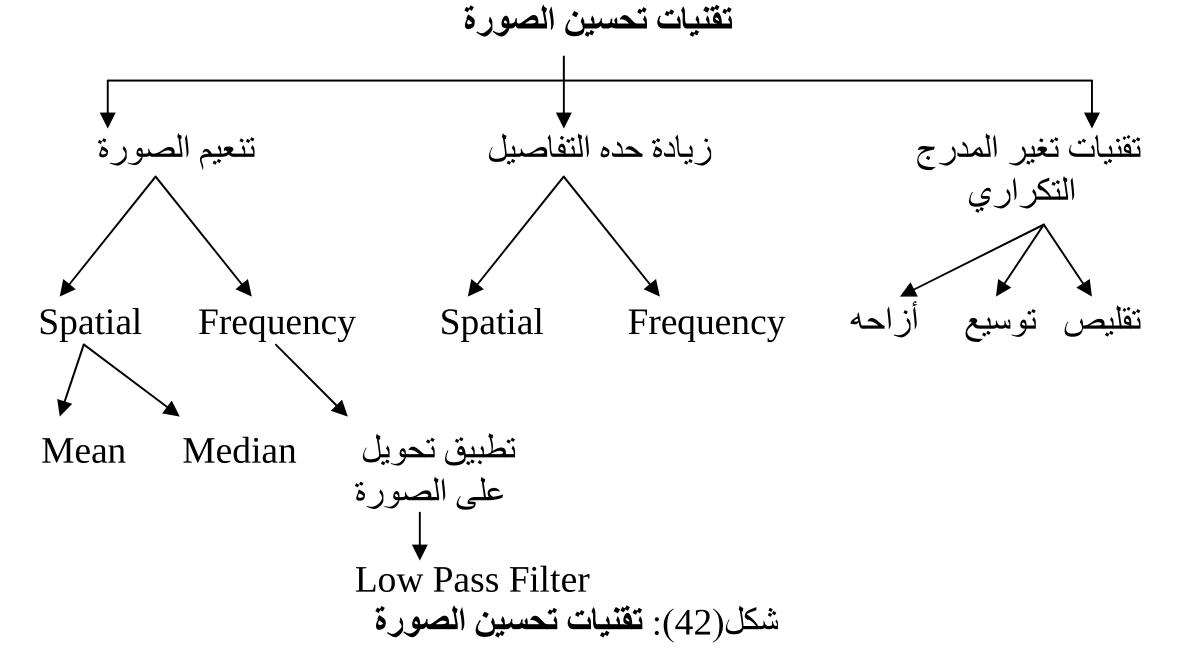
تستخدم هذه المرشحات بعد تطبيق أحد التحويلات على الصورة ثم تطبيق مرشحات الأحرار المنخفض التي تحتفظ بمعلومات الصورة ذات الترددات الواطئة وتهمل الترددات العالية وهذه المرشحات هي :-

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Low Pass Filter

المخطط العام تقنيات تحسين الصورة هو:



شكل(42): **تقنيات تحسين الصورة**

3-4 استرجاع (أعادة ترميم الصورة) Image Restoration :

وهي الطريقة التي يستخدم لإعادة ترميم واسترجاع الصورة بواسطة استخدام طريقة المعالجة عملية الاسترجاع مستخدمة موديل خاص

$$d(r,c) = h(r,c) * I(r,c) + n(r,c) \quad \ldots\ldots\ldots(19)$$

حيث * تمثل عملية الالتفاف Convolution

d (r, c) الصورة المسترجعة Degradation Image

h (r, c) عملية التحسين Degradation Process

I (r, c) هي الضوضاء التي تظهر في الصورة Noise Function

ملاحظة// الضوضاء هي أي معلومات غير مرغوب بها تظهر في الصورة .

مصادر الضوضاء التي تظهر في الصورة هي :-

* قناة الاتصال
* الضوضاء التي تظهر أثناء مرحلة استحصال الصورة Image acquisition التي تتم فيها تحويل الصورة من أشارة كهربائية مستمرة إلى الشكل الرقمي التي تتقبله الحاسوب .

أنواع الضوضاء Noise Type

أ- Gaussian Noise

التمثيل الرياضي لهذا النوع من الضوضاء هو

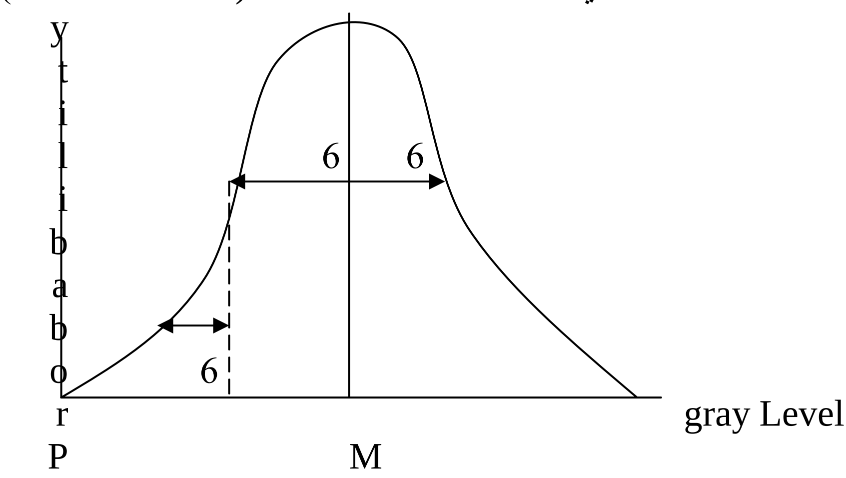
$$\text{Histogram Gaussian} = \frac{1}{\sqrt{\Pi\, 6^2}}\; e^{-(g-m)2/262} \qquad \ldots\ldots\ldots(20)$$

حيث

g هي القيمة اللونية gray level لعناصر الصورة

m المعدل

6^2 الانحراف المعياري standard deviation (6^2 = Variance)



شكل (43):توزيع الضوضاء من نوع Gaussian

ب- Uniform Noise

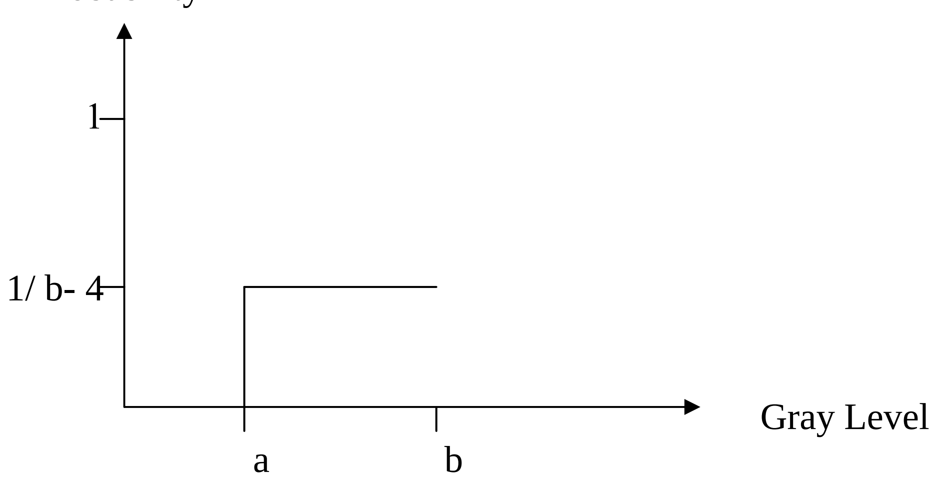
التمثيل الرياضي

$$\text{Histogram Uniform} = \begin{cases} \dfrac{1}{b-a} & \text{for } a \le g \le b \end{cases}$$

0 else where …………….(21)

$$\text{Mean} = \frac{a + b}{2} \quad \text{……………………………….(22)}$$

$$\text{Variance} = \frac{(b - a)^2}{12} \quad \text{…………………………….(23)}$$

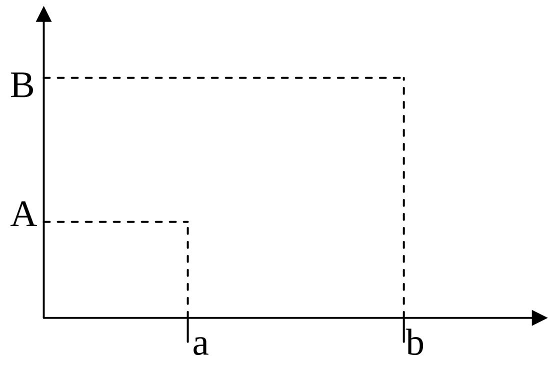


Uniform Noise:(44)شكل

ج- Salt & Popper noise

التمثيل الرياضي

Histogram Salt & Popper = A for g = a (Popper)
B for g = b (Salt) …….(24)



Salt & Popper noise:(45)شكل

4-4 تحسين الصورة حسب المجالات:

أن الهدف الرئيسي من تقنيات التحسين هو معالجة صورة معينة بحيث تكون النتيجة أكثر ملائمة من الصورة الأصلية لتطبيق محدد .

طرق المجال – ألحيزي :

أن المصطلح " المجال ألحيزي " يرجع إلى تجمعات العناصر التي تشكل صورة ما ، وطرق المجال – ألحيزي هي إجراءات تعمل مباشرة على هذه العناصر . يمكن التعبير عن دوال معالجة الصورة في المجال ألحيزي بالصيغة التالية :

$$X (x , y) = T [\int (x , y)] \text{ ....(25)}$$

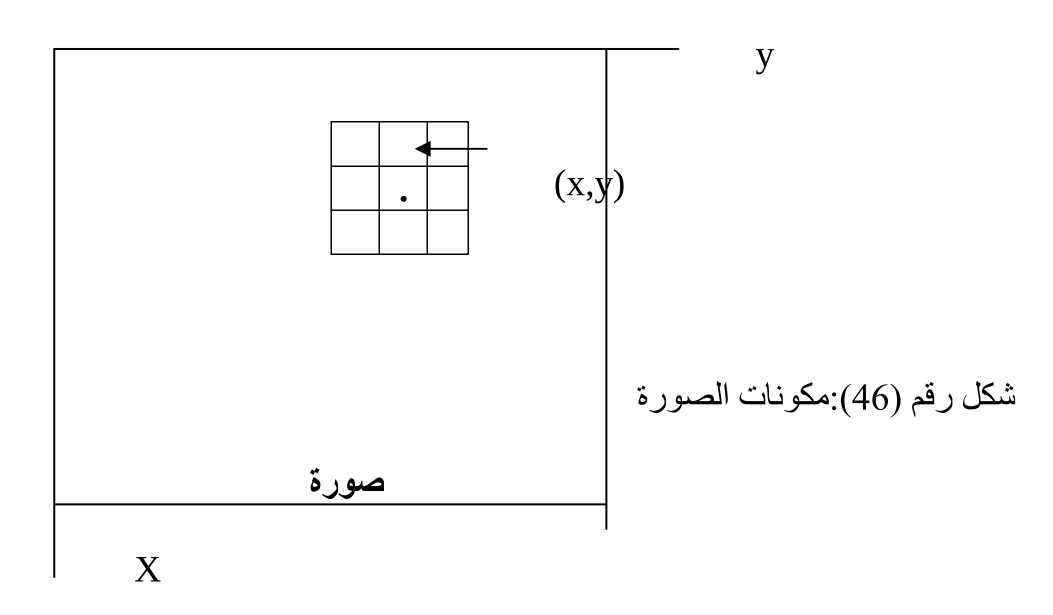
حيث أن :-

$\int (x , y)$ هي صورة الدخل

$X (x , y)$ هي صورة المعالجة

T هو عامل مؤثر على $\int$

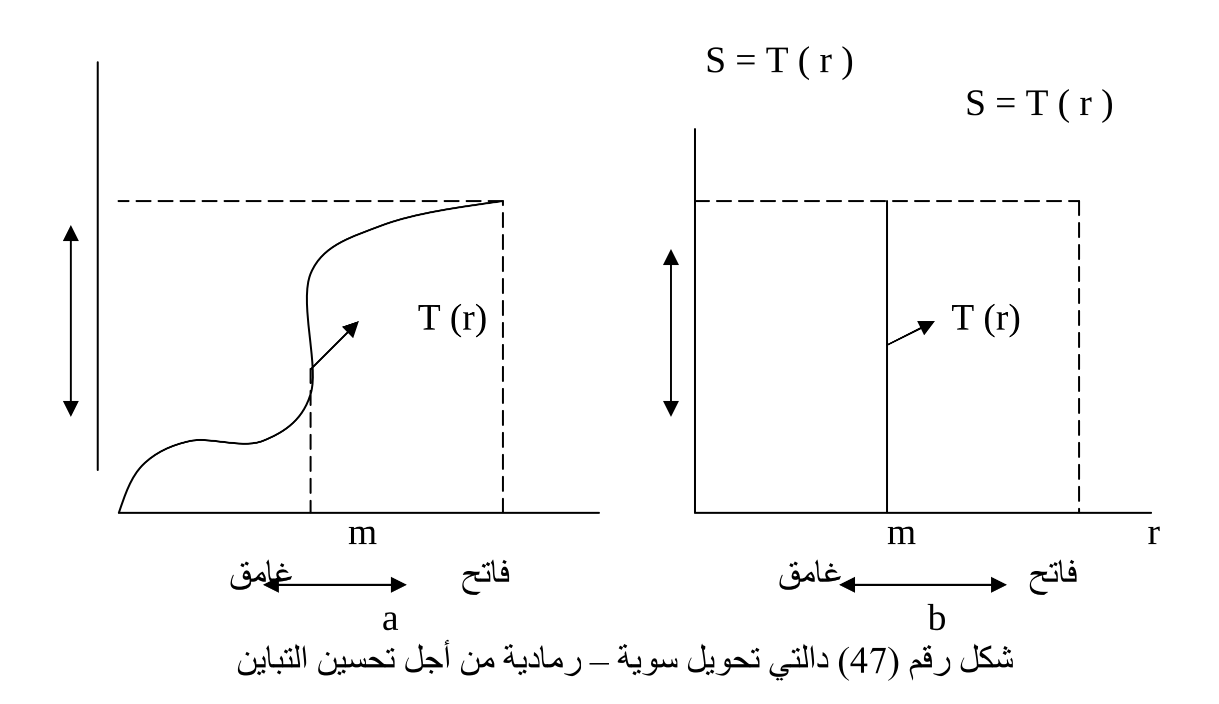
ومعرف في جوار ما ال(x , y) ومن الممكن أيضا أن نجعل T يؤثر على مجموعة من صور الدخل كما في حال إجراء جمع عناصر K صورة عنصرا فعنصرا من أجل تخفيض الضجيج يين



شكل رقم (46):مكونات الصورة

أن أبسط شكل لT هو عندما تكون أبعاد الجوار [x] . في هذه الحالة تعتمد **g** على قيمة $\int$ عند (x,y) فقط وتصبح T دالة – تحويل سويه – رمادية gray level transformation function (وتدعى أيضا " دالة نقل " Mapping Function) .

وأن هذا التحويل هو أنتاج صورة ذات تباين أعلى من تباين الصورة الأصلية وذلك تعتيم السويات الأقل من سوية ما m وزيادة لمعان السويات الأعلى من m في طيف العنصر الأصلي في هذه التقنية المعروفة بـ" مد التباين " Contrast Stretching " تضغط سويات r الاخفض من m بواسطة دالة التحويل إلى مدى ضيق من s باتجاه النهاية المعتمة من الطيف . أن الأثر المعاكس يحدث من أجل قيم r الأعلى من m .



شكل رقم (47) دالتي تحويل سوية – رمادية من أجل تحسين التباين

أن الجوارات الأكبر تسمح باستعمال دوال معالجة متنوعة تذهب إلى مجرد تحسين الصورة بغض النظر عن التطبيق و الأسلوب العام هو أن ندع قيمة ʃ في الجوار المحدد ل(x,y) يحدد قيمة g عند الاحداثيين .

أن أحد الأساليب الرئيسية في هذا التشكيل يبني على ما يسمى (مراشيح أو طبعات أو أقنعه) النافذة هي أساسا مصفوفة ثنائية الأبعاد (مثلا 3*3) .

أفترض أن لدينا صورة ثابتة وتحتوي نقاطا معزولة بشكل متباعد وشدتها تختلف عن شدة الخلفية يمكن كشف هذه النقاط.

أن أجراء المستخدم : يحرك مركز النافذة (المرقم بـ 8) على الصورة عند موضع كل عنصر في الصورة نضرب كل عنصر محتوى ضمن مساحة معامل النافذة المقابل أي العنصر الموجود في مركز النافذة يضرب بـ 8 في حين أن جيرانه الثمانية تضرب بـ 1- .

-1	-1	-1
-1	8	-1
-1	-1	-1

شكل رقم (48) نافذة لكشف نقاط معزولة تختلف عن خلفية ثابتة

أذا جعلنا $w_1 \dots w_9$ تمثل معاملات النافذة وأخذنا بعين الاعتبار الجيران الثمانية ل(x , y) يمكن أن تقوم بإنجاز العملية التالية :

$$T [\int (x , y)] = w_1 \int (x - 1 , y - 1) + w_2 \int (x - 1 , y) + w_3 \int (x - 1 , y + 1) + w_4 \int (x , y - 1) + w_5 \int (x , y) + w_6 \int (x , y + 1) + w_7 \int (x + 1 , y - 1) + w_8 \int (x + 1 , y) + w_9 \int (x + 1 , y + 1)$$

أذا أخرنا w_1-1/9 من أجل i-1,2,3…9 وجعلنا $8 (x ,y) - T [\int (r ,y)]$ ستكون قيم 8 عند (x , y) وجيرانه الثمانية .

W_1 (x – 1 , y – 1)	W_2 (x – 1 , y)	W_3 (x – 1 , y + 1)
W_4 (x , y – 1)	W_5 (x , y)	W_6 (x , y + 1)
W_7 (x + 1 , y – 1)	W_8 (x + 1 , y)	W_9 (x + 1 , y + 1)

شكل رقم (49) نافذة عامة أبعادها 3*3 معاملات ومواقع عناصر صورة

طرق المجال – الترددي :

أن أساس تقنيات المجال الترددي هو نظرية الطي . لتكن g (x , y) صورة شكلت بطي صورة (x ,y) ∫ ومؤثر h(x , y) غير تابع للموضوع Position Invariant أي

$$\mathbf{S (x , y) = h (x, y) * \int (x ,y)} \qquad \text{..............................(26)}$$

عندئذ ومن نظرية الطي فأن علاقة المجال الترددي التالية صحيحة :

$$\mathbf{G (u , p) = H (u , p) F (u , p)} \qquad \text{............(27)}$$

حيث أن :-

F , H , G هي تحويلات فورية لg وh وf على التوالي . أن التحويل H (u , v) يدعى عادة دالة التحويل Transfer Function للعملية .

مسائل تحسين الصورة يمكن أن يعبر عنه في تطبيق تحسين صورة نموذجي ، وتعطي f(x,y) ويتم بعد حساب F(u,v)

اختيار H(u ,v) بحيث تبرز الصورة المرغوبة والمعطاة بالعلاقة التالية :

$$\mathbf{g (x , y) = \zeta^{-1} \{ H(u, v) F (u, v) \}} \quad ......(28)$$

خاصة ما للصور f(x , y) فعلى سبيل المثال يمكن أن تبرز الحواف في f(x , y) باستعمال دالة H(u, v) تؤكد على الترددات العالية لتحويل فورييه F (u, v) .

ليكن المتحول r يمثل السوية الرمادية للعناصر الموجودة في الصورة والتي نزيد تحسينها وللتبسيط سنفترض أن قيم العناصر قد قيست بحيث تقع جميعها في المدى .

$$\mathbf{0 \le r \le 1}$$

حيث أن r-0 تمثل الأسود و r-1 تمثل الأبيض على السلم الرمادي من أجل أيه r في البون [0,1] نحصل على:

$$\mathbf{S = T (r)} \quad .... (29)$$

والتي تنتج سوية S لكل عنصر r في الصورة الأصلية . يفترض أن دالة التحويل المعطاة تحقق الشرطيين التاليين :

1) T(r) وحيدة القيمة و تتزايد على وتيرة واحده Monotonically في البون $0 \le r \le 1$

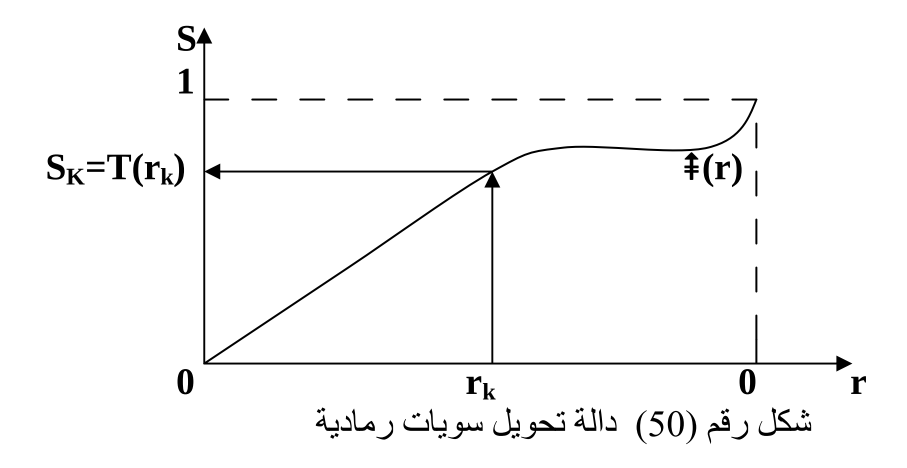
2) $0 \le T (r) \le 1 \quad \text{for} \quad 0 \le r \le 1$

يحفظ الشرط (1) الترتيب من الأسود إلى الأبيض على السلم الرمادي ، في حين ضمن الشرط (2) رسما Mapping منسجما مع المدى المسموح به لقيم عناصر الصورة الذي يوضح دالة تحويل تحقيق هذين الشرطين
سنعبر عن التحويل العكسي من S رجوعا إلى r بالشكل التالي

$$\mathbf{r = T^{-1}(s) \qquad 0 < S < 1} \quad ....(30)$$

حيث نفترض أن $T^{-1}(s)$ يحقق أيضا الشرطين (1) و (2) بالنسبة للمتحول s .

أن السويات الرمادية في صورة ما هي كميات عشوائية في البون [0,1] . فبافتراض أنها متحولات مستمرة ، يمكن أن توصف السويات الرمادية الأصلية والمحولة ، بدالتي كشافة احتمالها $p_r(r)$ و $p_s(s)$ على التوالي يمكن أن يقال الكثير حول



شكل رقم (50) دالة تحويل سويات رمادية

أن صورة لمستوياتها الرمادية دالة كثافة تستخدم مناطق مضيئة (فاتحة) لأن غالبية عناصرها رمادية فاتحة .

أننا نحتاج $T(s)$ فقط لتسوية الهيستوغرام بأن نثبت أن الكثافة ناتجة $p_s(s)$ هي في الحقيقة منتظمة لأن هذا يتطلب الحصول على $T^{-1}(s)$ في الواقع العملي هذه الخطوة غير مطلوبة لأن الدالة مستقلة عن دالة التحويل العكسي .

أن الحل من أجل r بدلالة s يعطي

$$r = T^{-1}(s) = 1 \pm \sqrt{1-s} \quad \ldots\ldots(31)$$

لأن r تقع البون [0,1] فأن الحل التالي فقط صحيح :-

$$r = T^{-1}(s) = 1 - \sqrt{1-s} \qquad \ldots\ldots\ldots(32)$$

يحصل على دالة كثافة الاحتمال لـs بتعويض النتائج:

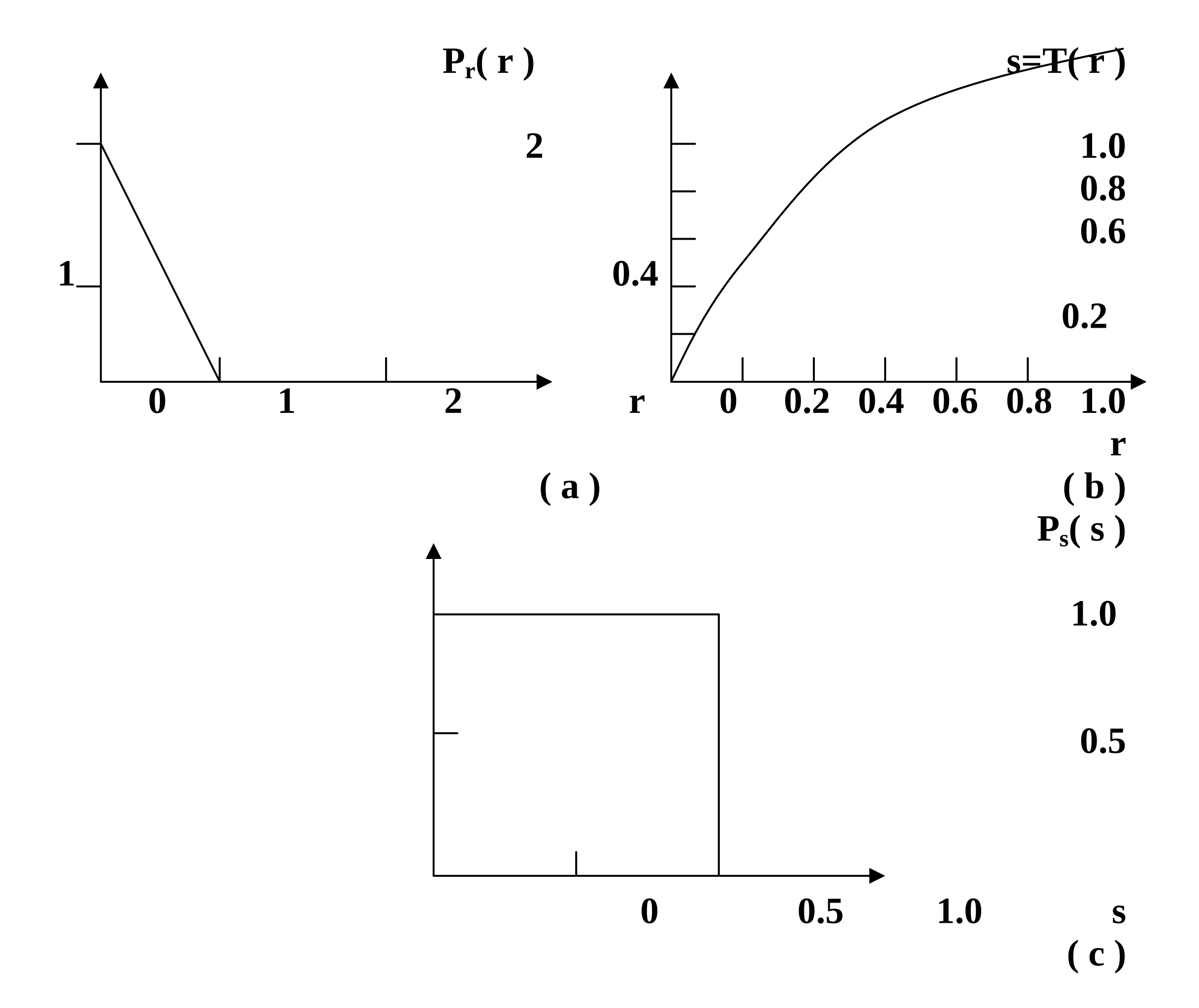
$$Ps(s) = [P_r(r)\, dr / ds]_{r = T-1(s)} \ldots\ldots\ldots\ldots(33)$$

$$= [(-2r+2)\, dr / ds]_{r = 1 - \sqrt{1-s}} \quad \ldots\ldots\ldots\ldots(34)$$

$$= [(-2\sqrt{1-s})\, d / ds\, (\sqrt{1-s})] = 1 \qquad 0 \le s \le 1 \ldots\ldots(35)$$

التي هي كثافة منتظمة في البون المرغوب .
من أجل السويات الرمادية التي تأخذ قيما متقطعة نتعامل مع احتمالات تعطي بالعلاقة التالية :

$$P_r(r_k) = n_k / n \qquad 0 \le r_k \le 1$$
$$K = 0, 1, \ldots\ldots\ldots\ldots, L-1$$
$$\ldots\ldots\ldots\ldots(36)$$



شكل رقم (51) توضح طريقة تحويل الكثافة المنتظمة (a) دالة كثافة الاحتمال الأصلي . (b) دالة التحويل . (c) الكثافة المنتظمة الناتجة .

حيث أن L هي عدد السويات و($p_r(r_k)$ هي احتمال السوية الرمادية رسم K و n_k هي عدد المرات التي تظهر فيها هذه السوية في الصورة و n هي العدد ألأجمالي للعناصر في الصورة .

أن الرسم البياني $P_1(r_k)$ مقابل r_k يدعى عادة الهيستوغرام histogram والتقنية المستخدمة للحصول على هيستوغرام منتظم تدعى تسوية الهيستوغرام Histogram Equalization أو Histogram Linearization .

أن الشكل المتقطع يعطي بالعلاقة التالية :

$$S_k=T(r_k)=\sum_{J=0}^{0} n_k / n \quad ....(37)$$

$$= \sum_{J=0}^{k} P_r(r_k)$$

$$0 \le r_k \le 1$$

$$K = 0 , 1 , ..., L - 1$$

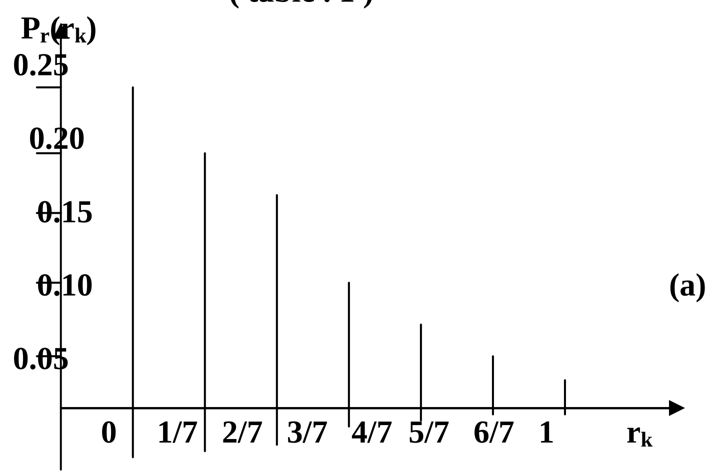
ويشار إلى التحويل العكسي كما يلي

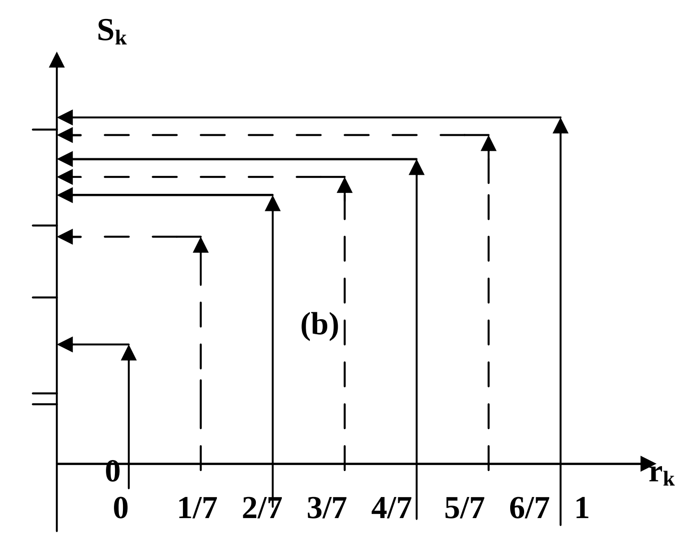
$$r_k = T^{-1}(s_k) \text{ ....} (38)$$
$$0 \le s_k \le 1$$

مثال // أفترض أن صورة أبعادها 64*64 وفيها 8 سويات رمادية لما توزيع السويات الرمادية

r_k	n_k	$P_r(r_k) = n_k/n$
$r_0 = 0$	790	0.19
$r_1 = 1/7$	1023	0.25
$r_2 = 2/7$	850	0.21
$r_3 = 3/7$	656	0.16
$r_4 = 4/7$	329	0.08
$r_5 = 5/7$	245	0.06
$r_6 = 6/7$	122	0.03
$r_7 = 1$	81	0.02


(table . 1)



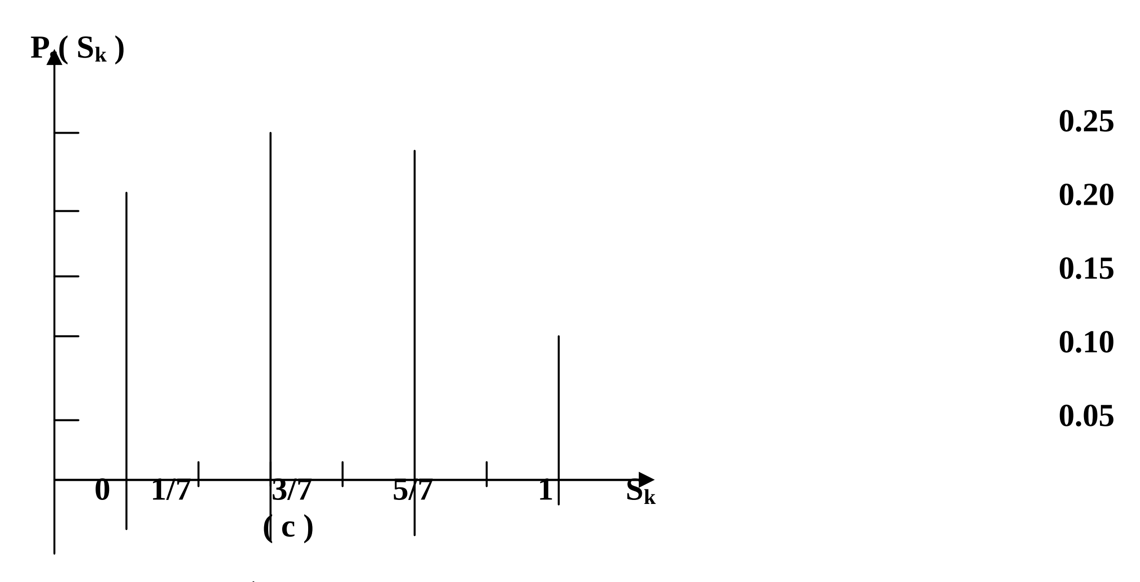


1.00

0.80

0.60

0.20



الشكل رقم (52) يوضح طريقة تسوية الهيستوغرام (a) الهيستوغرام الأصلي (b) الهيستوغرام المسوي(c)

يحصل دالة التحويل :

$$S_0 = T(r_0) = \sum_{J=0}^{0} P_r(r_J)$$
$$= P_r(r_0)$$
$$= 0.19$$

بشكل مشابه

$$S_1 = T(r_1) = \sum_{J=0}^{1} P_r(r_J)$$
$$= P_r(r_0) + P_r(r_1)$$
$$= 0.44$$

$S_2 = 0.65$ $S_5 = 0.95$
$S_3 = 0.81$ $S_6 = 0.98$
$S_4 = 0.89$ $S_7 = 1.00$

يجب أن تنسب كل قيمة من القيم المحولة لأقرب سوية صحيحة كالأتي:

$S_0 \cong 1/7$ $S_1 \cong 3/7$ $S_2 \cong 5/7$
$S_3 \cong 6/7$ $S_4 \cong 6/7$ $S_5 \cong 1$
$S_6 \cong 1$ $S_7 \cong 1$

يلاحظ أن يوجد خمس سويات هيستوغرام – مسوي رمادية مميزه فقط . و بإعادة تعريف التعابير الرياضية لتأخذ هذا الأمر بالحسبان ، تنتج السويات التالية :

$S_0 = 1/7$ $S_1 = 3/7$ $S_2 = 5/7$
$S_3 = 6/7$ $S_4 = 1$

ولأن $r_0–0$ كانت قد حولت (Mapped) إلى $S_0–1/7$ يوجد 790 عنصرا محولا بهذه القيمة الجديدة . ويوجد أيضا 1023 عنصرا بالقيمة $S_1–3/7$ و 850 عنصرا بالقيمة $S_2–5/7$ ولكن لأن كلا من السويتين r_3 و r_4 كانت قد حولت إلى $S_3–6/7$ ويوجد 556 ،329 -985 عنصرا بهذه القيمة الجديدة بشكل مشابه ، يوجد 245 ،122 ،81 -844 عنصرا بالقيمة $S_4–1$. أن تقسيم هذه الأعداد على 4096 يعطي للهيستوغرام. ولأن الهيستوغرام تقريب لدالة كثافة الاحتمال ينـدر الحصول على نتائج مسطحة تماما عند التعامل مع السويات المتقطعة .

5-4 التوصيف المباشر للهيستوغرام Direct Histogram Specification:

أن الطريقة السابقة مفيدة جدا لكنها ليست ملائمة من أجل تطبيقات تحسين الصور التفاعلي Intensive Image Enhancement لأن إمكانات هذه الطريقة محدودة بتوليد نتيجة واحدة فقط ، تقريب لهيستوغرام منتظم .

$$S = T (r) = \int_0^r p_r (w) dw …….(39)$$

أذا كانت الصورة المرغوبة متوفرة فأن مستوياتها الرمادية يمكن أن تسوى باستعمال دالة التحويل التالية :-

$$P = G (z) = \int_0^z p_z (w) dw ……(40)$$

أن العملية العكسية $z\text{-}G^{-1} (v)$ ستعيد عندئذ السويات المرغوبة . تشكيل افتراضي لأن سويات z هي بالضبط ما نجرب الحصول علية . ولكن أن $p_s (s)$ و $p_v (v)$ ستكونان كثافتين منتظمتين متماثلتين لأن النتيجة النهائية مستقلة عن الكثافة داخل التكامل .

اذا استعملنا المنتظمة التي حصلنا عليها من الصورة الأصلية بدلا من v في العملية سيكون للسويات الناتجة $z\text{-}G^{-1}(s)$ دالة الكثافة المرغوبة . بافتراض أن وحيد القيمة يمكن تلخيص الأجراء كما يلي :-

1- مستويات الصورة الأصلية
2- حدد دالة الكثافة المرغوبة واحصل على دالة التحويل G(z)
3- طبق دالة التحويل العكسي $z\text{-}G^{-1}(s)$ على السويات التي حصلت عليها في الخطوة 1 .

هذا الأجراء يعطي النسخة المعالجة للصورة الأصلية حيث أن السويات الجديدة مميزة بالكثافة الجديدة $p_z (z)$.

أن طريقة تحديد الهيستوغرام تستخدم دالتي تحويل T (r) متبوعة بمسألة بسيطة أن ندمج كلا من خطوتي التحسين في دالة واحدة تعطي النتائج المرغوبة بدءا من عناصر الصورة الأصلية .

$$Z = G^{-1}(s) ………..(41)$$

نحصل على دالة التحويل المدموج

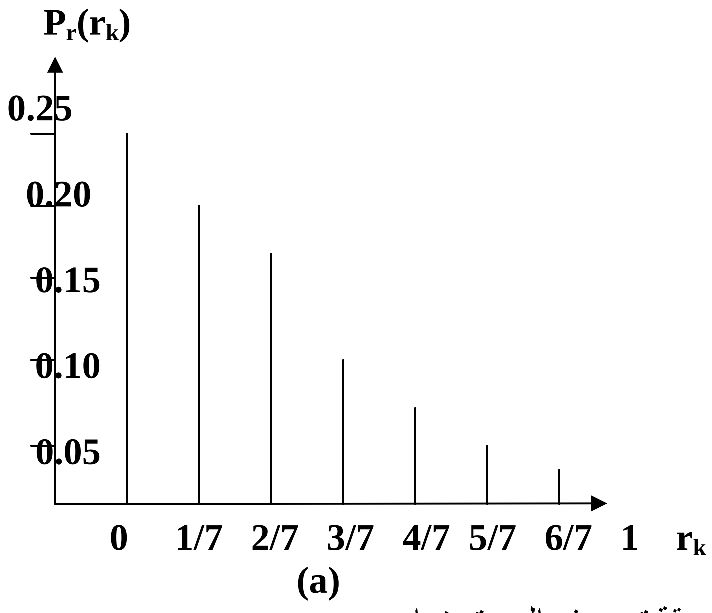
$$Z = G^{-1} [T (r)]…………..(42)$$

الذي يربط الـr بالـz يلاحظ أنه عندما تكون $T(r)-[T(r)]^{-1}G$ تختصر هذه إلى تسوية الهيستوغرام .

مثال // اذا كانت لديك قيم الصورة التي فيها 8 – سويات رمادية وأبعادها 64*64 . استخدم القوانين التي تحدد دالة التحويل:

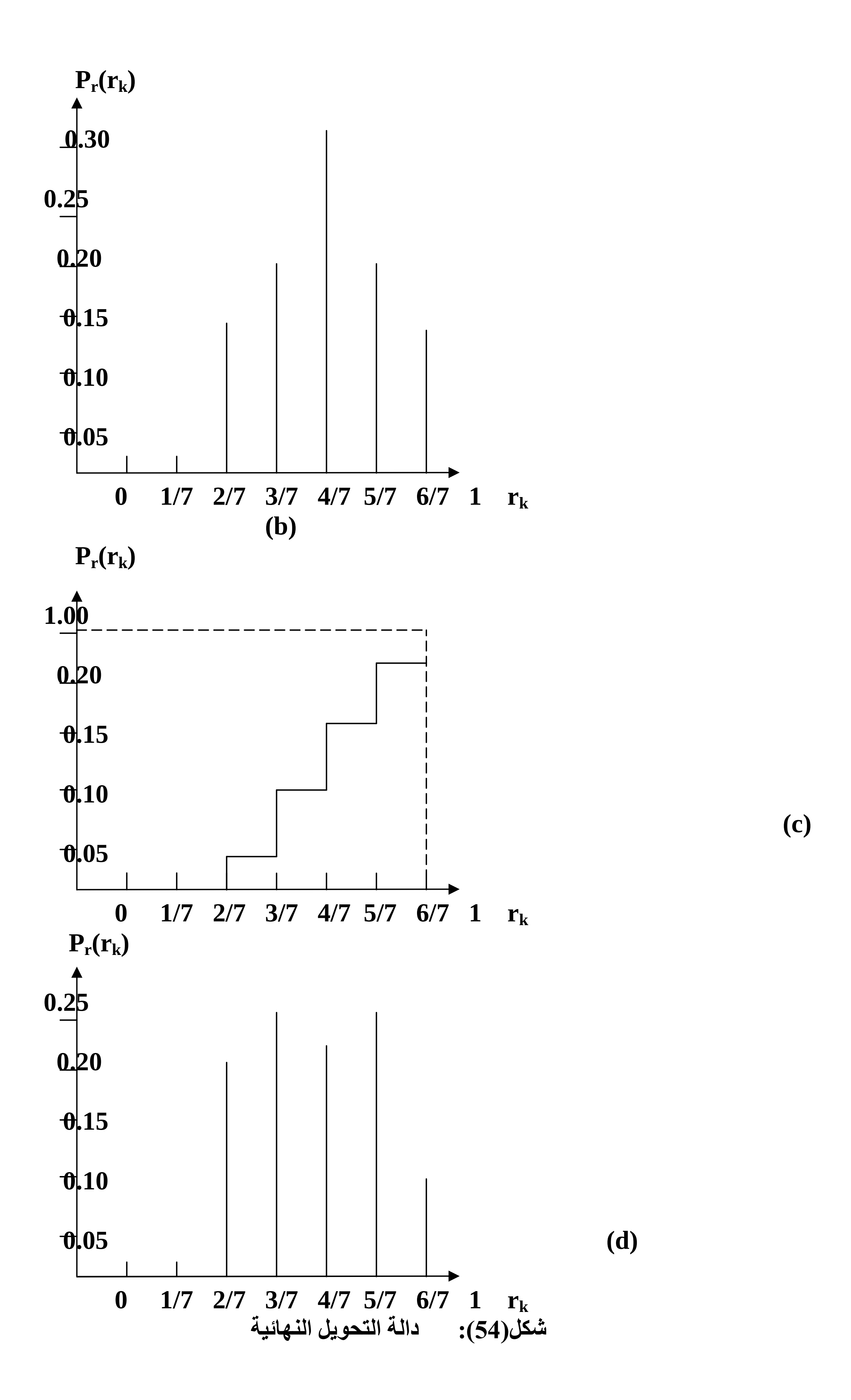
z_k	$P_z(z_k)$
$z_0 = 0$	0.00
$z_1 = 1/7$	0.00
$z_2 = 2/7$	0.00
$z_3 = 3/7$	0.15
$z_4 = 4/7$	0.20
$z_5 = 5/7$	0.30
$z_6 = 6/7$	0.20
$z_7 = 1$	0.15

(table . 2)



(a)

في هذا الشكل رقم (53) توضيح طريقة توصيف الهيستوغرام
(a) هيستوغرام أصلي
(b) هيستوغرام موصف
(c) دالة تحويل
(d) الهيستوغرام الناتج



شكل(54): دالة التحويل النهائية

أن الخطوة الأولى في الأجراء هي الحصول على التحويلات Mapping تسوية – الهيستوغرام بعد ذلك نحسب دالة التحويل

$$r_k = G(z_k) = \sum_{k=0}^{k} p_s(z_k) \quad \text{.........(43)}$$

وهذا يعطي القيم التالية :

$r_0 = G(z_0) = 0.00$ **$r_4 = G(z_4) = 0.35$**
$r_1 = G(z_1) = 0.00$ **$r_5 = G(z_5) = 0.65$**
$r_2 = G(z_2) = 0.00$ **$r_6 = G(z_6) = 0.85$**
$r_3 = G(z_3) = 0.15$ **$r_7 = G(z_7) = 1.00$**

$r_k \rightarrow s_k$		n_k	$P_s(s_k)$
r_0 →	$s_0 = 1/7$	790	0.19
r_1 →	$s_1 = 3/7$	1023	0.25
r_2 →	$s_2 = 5/7$	850	0.21
r_3, r_4 →	$s_3 = 3/7$	985	0.24
r_5, r_6, r_7 →	$s_4 = 4/7$	448	0.11

(table . 3)

للحصول على قيم z نطبق معكوس لتحويل G على السويات s_k المبينة بطريقة تسوية الهيستوغرام ولأننا نتعامل مع قيم منفصلة يجب عادة أن نجري تقريبا في التحويل Mapping العكسي مثلا أن أقرب نظير لـ0.14 $s_0 = 1/7 =$ أو $G(z_3)=0.15$ أو استعمال التحويل العكسي $G^{-1}(0.15)z_3$ وهكذا فأن s_0 حولت إلى سوية z_3 أن استعمال هذا الأجراء يعطي تحويلات التالية :-

$s_0 = 1/7 \rightarrow z_3 = 3/7$
$s_1 = 3/7 \rightarrow z_4 = 4/7$
$s_2 = 5/7 \rightarrow z_5 = 5/7$
$s_3 = 6/7 \rightarrow z_6 = 6/7$
$s_4 = 1 \rightarrow z_7 = 1$

يمكن دمج هذه النتائج لإعطاء التحويلات Mapping المباشرة التالية :-

$r_0 = 0 \rightarrow z_3 = 3/7$ $r_4 = 4/7 \rightarrow z_6 = 6/7$
$r_1 = 1/7 \rightarrow z_4 = 4/7$ $r_5 = 5/7 \rightarrow z_7 = 1$
$r_2 = 2/7 \rightarrow z_5 = 5/7$ $r_6 = 6/7 \rightarrow z_7 = 1$

$r_3 = 3/7 \longrightarrow z_6 = 6/7$ $r_7 = 1 \longrightarrow z_7 = 1$

أن أعادة توزيع عناصر الصورة وفقا لهذه التحويلات والتقسيم على الهيستوغرام

r_k	n_k	$P_z(z_k)$
$r_0 = 0$	0	0.00
$r_1 = 1/7$	0	0.00
$r_2 = 2/7$	0	0.00
$r_3 = 3/7$	790	0.12
$r_4 = 4/7$	1023	0.25
$r_5 = 5/7$	850	0.21
$r_6 = 6/7$	985	0.24
$r_7 = 1$	448	0.11

(table . 4)

الفصل الخامس

ضغط الصورة الرقمية

1-5 ضغط الصور Image Compression:

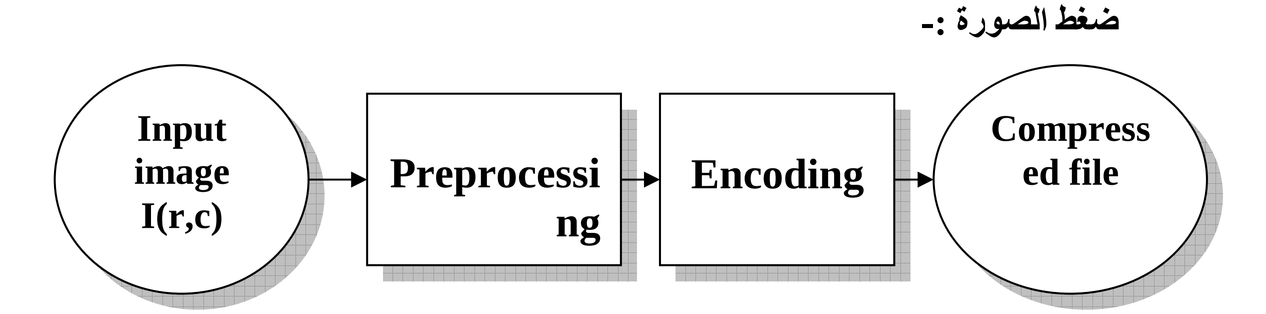
ضغط الصور هو تقليل حجم بيانات الصورة مع الاحتفاظ بالبيانات الضرورية لفايل الصورة أي أن حجم الفايل المتقلص يسمى بالملف المضغوط الذي يستخدم لاسترجاع الصورة المضغوطة يجب معرفة :-

1- نوع الصورة Type
2- حجم الصورة Size
3- بيانات الصورة Data

الرسم التالي يوضح عملية الضغط التي تتكون من مرحلتين

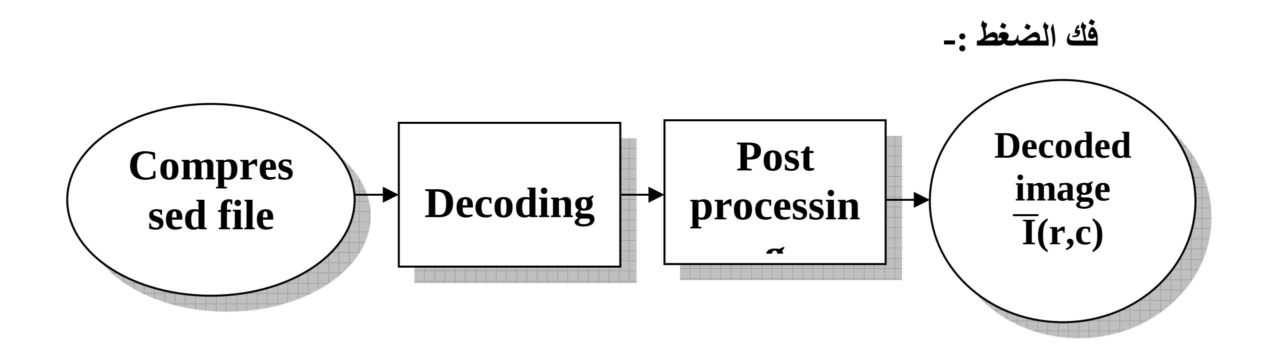
1- ضغط الصورة
2- فك الضغط

ضغط الصورة :-



a : compression

فك الضغط :-



b : Decompression

شكل(55):ضغط صورة وفك الضغط

2-5 نسبة الضغط Compression ratio :

هي تحديد حجم الفايل المضغوط أي مقدار ضغطة وتأتي من القانون التالي :-

Compression ratio = uncompressed file size / compressed file size
= size u / size c ……………..(44)

$$\text{نسبة الضغط} = \frac{\text{حجم الصورة غير المضغوطة}}{\text{حجم الصورة الأصلية}}$$

مثال // لديك صورة حجمها 256 (8 bits) لكل عنصر وبعد عملية الضغط أصبح حجمها (6.554) بايت . أحسب نسبة الضغط ؟؟

Size u / Size c = 256 * 256 / 6554 = 65536 / 6554 = 9.999 ≅ 10
as **10 : 1**

توجد طريقة ثانية لإيجاد نسبة الضغط اعتمادا على الـbit Per Pixel التي تساوي القانون التالي

CR = Bits Per Pixel = number of Bits / number of Pixel ….(45)

= (8) (number of byte) / N*N

طبق المثال السابق على هذا القانون

CR = Bits Per Pixel = 6554 * 8 / 256 * 256 = 52432 bit / 65536
= 0.8 Bit Per Pixel

الفرق بين القانونين

1- الصورة قبل الضغط توضع في البسط لكي يتم تحويلها إلى بتأت بضربها * 8 .
2- الطريقة الثانية نسبة الضغط أعلى أو أكبر من السابقة .

أما مساوئ الطريقة الأولى فهي
عند ضغط البايت لكل بكسل فيؤدي إلى دمج الألوان لكل نهاية البايت الأول وبداية البايت الثاني .

من مساوئ الطريقة الثانية
تتعامل مع بت لكل بكسل في كل مرة والبت عبارة عن خلية بينما الكسل هو عبارة عن 8 خلايا وبذلك في كل مرة تفقد 7 خلايا فيؤدي إلى التشوية .

مثال // أذا كان لديك صورة من نوع BGR لونية سعتها 512*512 و24 بت لكل بكسل (أي 8 بت لكل لون) .. المطلوب إيجاد نسبة الضغط بالأوقات (الدقائق) علما أن الوقت المستغرق للضغط كان (28.8Kboud) يعني (Kilo bits / Second)

الحل // بما أن القوانين المستخدمة لا تحتوي على الأوقات لذا بجب تكوين قانون جديد يتمثل بالوقت كتالي :-

$$\frac{\textbf{(512*512 pixels) (24 bits / pixel)}}{\textbf{(28.8*1024 bits / second)}} \approx \textbf{213 second} \approx \textbf{3.6 minutes}$$

نقوم بتحويل البت إلى ثواني Second والثواني نحولها إلى دقائق
ال Minuets هي 60 ثانية

= 213 / 60 = 3.6 Minuets

3-5 معايير الدقة أو مقاييس التقييم (الموثوقيه) FIDELITY CRITERIA:

تتطلب خوارزميات الضغط (ضغط الصور) تحديد أقل كمية من البيانات لتمثيل الصورة كمعلومات ضرورية بدون التأثير عليها وإيجاد مقاييس :

1- مقياس الهدفobjective fidelity criteria

2- إيجاد نسبة الإشارة إلى الضوضاء (Single to noise ratio (SNR

1 – مقياس الهدف : تستخدم هذه المقاييس لإيجاد معاملات نسبة الخطاء بين الصورة الأصلية والصورة المسترجعة

1- نجد الجذر ألتربيعي لمعدل مربعات الخطاء root – mean – square error الذي يتمثل بتحديد الفرق الكلي للخطاء بين الصورة الأصلية والصورة المسترجعة .
ويوجد قانون لإيجاد الخطاء للجذر ألتربيعي :-

$$\textbf{Total error} = \sum_{r=0}^{n-1}\sum_{c=0}^{m-1} [\, \bar{I}(r,c) - I(r,c) \,] \ldots\ldots\ldots(46)$$

حيث أن :-
I(r,c) هي الصورة الأصلية .
Ī(r,c) هي الصورة المسترجعة بعد الضغط .

n ,m هي سعة الصورة في حالة أذا كانت الصورة مربعه فتصبح n*n أما أذا كانت غير مربعة فأنها سوف تكون n*m .

يمكن أن نحصل على نسبة الجذر ألتربيعي للمعدلات مربعات الخطاء وكتالي (أذا كانت المصفوفة مربعة أو غير مربعة)

$$\textbf{RMSE} = \sqrt{1/N^2 \sum_{r=0}^{n-1}\sum_{c=0}^{n-1} [\ \bar{I}(r,c) - I(r,c)\ ]^2} \ldots\ldots\ldots(47)$$

معنى ذلك أن القيمة مربعات نسبة الخطاء كلما كانت نوعية كلما كانت الصورة المسترجعة جيدة إذاً نسبة الخطاء في الصورة كانت صغيرة أو قليلة كلما كانت أفضل لأن معلومات الصورة الأصلية لا تفقد بشكل كبير على عكس أذا كانت نسبة الخطاء كبيرة .

2 – نسبة الإشارة إلى الضوضاء :

تعد هذه النسبة الأكثر استخداما من الطريقة السابقة حيث أن هذا المقياس يمكنه تمثيل الصورة المسترجعة من خلال الإشارة والفرق بينها (الفرق بين الصورة الأصلية والصورة المسترجعة) وهنالك قانون لتمثيل ذلك :-

$$\textbf{SNR} = \sqrt{\frac{\sum_{r=0}^{n-1}\sum_{c=0}^{n-1} [\ \bar{I}(r,c)\ ]^2}{\sum_{r=0}^{n-1}\sum_{c=0}^{n-1} [\ \bar{I}(r,c) - I(r,c)\ ]^2}} \ldots\ldots\ldots\ldots(48)$$

مثال // جد مقياس الهدف و نسبة الإشارة إلى الضوضاء بين الصورتين I الصورة الأصلية و $\bar{I}$ الصورة المسترجعة ؟

$$I = \begin{bmatrix} 9 & 3 & 2 \\ 4 & 5 & 8 \\ 11 & 12 & 20 \end{bmatrix} \qquad \bar{I} = \begin{bmatrix} 18 & 2 & 1 \\ 5 & 4 & 7 \\ 11 & 12 & 20 \end{bmatrix}$$

الحل//

$$\text{RMSE} = \sqrt{1/3*3\ [\ (18-9) + (2-3) + (1-2) + (5-4) + \ldots + (20-20)\ ]^2}$$

$$= \sqrt{1/9\ [\ 9 - 1 - 1 + 1 - 1 - 1\ ]^2} = \sqrt{1/9 * [\ 6\ ]^2} = \sqrt{1/9 * 36} = \sqrt{4}$$

أما في حالة SNR هو أفضل وأكثر استخداما لأنه في قانونه هو التربيع الذي سوف يتخلص من القيمة السالبة فيكبر النسبة

$$\text{SNR} = \sqrt{\frac{[\ 18 + 2 + 1 + 5 + 4 + 7 + 11 + 12 + 20\ ]^2}{36}} = \sqrt{\frac{6400}{36}}$$

$$= \sqrt{177.7} = 13.33$$

يوجد مقياس أخير لطريقة الهدف Beak Signal to Noise Ratio يحسب قيمة الإشارة إلى الضوضاء حسب المعادلة التالية :-

$$PSNPR = 10 \log_{10} \frac{(L-1)^2}{1/N^2 \sum_{r=0}^{n-1} \sum_{c=0}^{n-1} [\,\overline{I}(r,c) - I(r,c)\,]^2} \quad \ldots\ldots\ldots(49)$$

حيث أن :-

L هي قيمة معينة من الصفر إلى 255

والL هي القمة وتعني أعلى رقم بالضوضاء بالصورة (أذا أعطى القمة نستخدم طريقة PSNPR بالهدف) .

2-المقاييس الموضوعية الشخصية subjective fidelity criteria

أن هذه الطريقة بسيطة البرمجة وتعتمد على وسائل بسيطة لتقييم الصورة المسترجعة أي أنها تعتمد على النظام البصري لدى الإنسان (المشاهدة) هنا يمكن أن تتمثل بالملاحظتين التاليتين :-

1- أن الأخطاء في المناطق المعتمة تكون ملحوظة أكثر من الأخطاء بالمناطق المضاءة .
2- الأخطاء في حواف الصورة تكون ملحوظة أكثر من خلفية الصورة .

بعد ذلك تعرض على أشخاص ويؤخذ المعدل لتقييم هذه الاستخلاص

أسباب الملاحظتين

1- لأن اللون الأسود أو المعتم هو من الألوان التي يبرز فيها أي لون أخر .

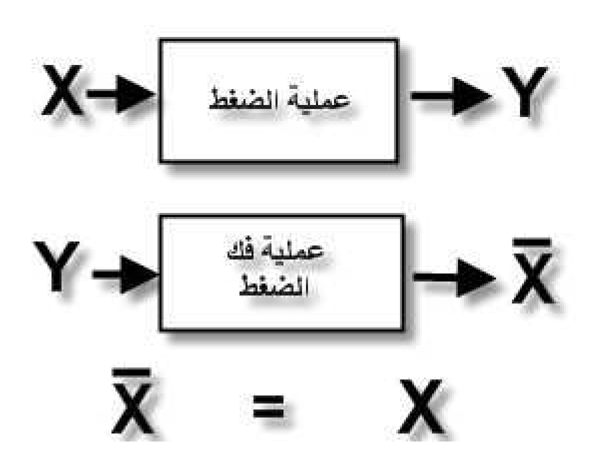
2- في الملاحظة الثانية وذلك لوجود فقدان قيم خاصة بالحافة سوف يؤثر على ما يأتية من الالوان المصاحبة لحافة لان:

A- بعض الأرقام هنا مجموعها مع الماسك لا يساوي واحد
B- الفرق بين القيم سوف يكون أكبر بحيث يؤثر على فصل القيم الكبيرة الحجم عن الصورة وبالتالي تفقد الصورة ملامحها الأصلية (الصفات الأصلية لها) .

4-5 طرائق الضغط للصورة الرقمية:

توجد طريقتين للضغط
1- طريق الضغط بدون فقدان قيم أي بياناتLossless data compression

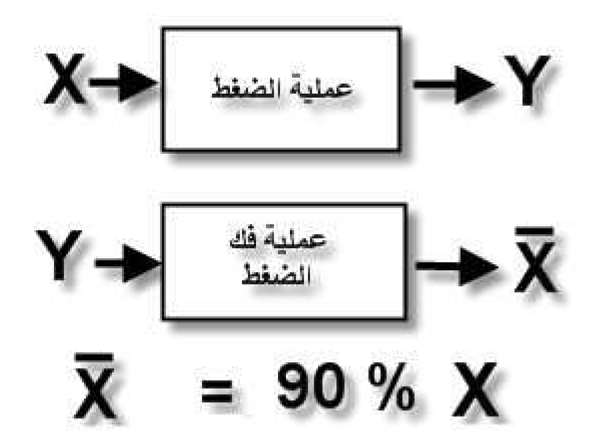
في هذا النوع لابد أن يكون الملف المضغوط –بعد فكه- مطابق تماما للملف الأصلي ، أي أنه لا يوجد فقد في المعلومات و من هنا جاءت التسمية، وهذا النوع يجب أن نستخدمه مع ملفات مثل الملفات التنفيذية ,EXE الملفات النصية ....TXT, DOC الخ.



شكل(56): طريق الضغط بدون فقدان قيم أي بيانات**Lossless data compression**

2- طريقة الضغط بفقدان قيم بياناتLossy data compression

أي أن الملف المضغوط عند فك ضغطه لن نحصل منه علي نسخة تكون مطابقة للملف الأصلي تماما و لكن سنحصل مثلا علي 90 %أو 80% منه بحيث يكون لدينا المعلومات المهمة عنه فقط أي أننا سنحصل علي ملف مشابه للملف الأصلي ولكن جودته تكون أقل من جودة الملف الأصلي و لعلكم تعرفون الفرق في الوضوح والحجم بين ملفات الصور ذات الامتداد BMP-الغير مضغوط-و الملفات ذات الامتداد GIF– المضغوط- ، وهذا النوع من الضغط مناسب تماما لملفات الملتيميديا كملفات الصوت والصورة والفيديو ومثال علي أنواع هذه الملفات ملفات ال GIF و ال JPEG و في الصوت مثلا Real Media MP3 , و مثال علي ملفات الفيديو Asf , Wmv.ويتم إستخدام هذا الأسلوب عند الرغبة في الحصول علي نسبة ضغط عالية جدا وليست هناك حاجه ضرورية لأن يكون الملف الناتج بعد عملية الضغط مطابق تماما للملف الأصلي.



شكل(57): طريقة الضغط بفقدان قيم بيانات Lossy data compression

في الطريقة الأولى تبقى نوعية الصورة جيدة ولكن نسبة ضغط قليلة أما بالطريقة الثانية فهي نوعية الصورة غير جيدة ولكن الضغط فيها كبير .



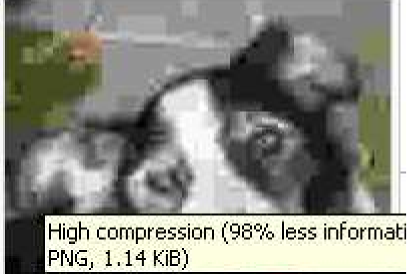
Original Image (lossless PNG, 60.1 KiB size) — uncompressed is 108.5 KiB



Medium compression (92% less information than uncompressed PNG, 4.82 KiB)



Low compression (84% less information than uncompressed PNG, 9.37 KiB)



High compression (98% less information than uncompressed PNG, 1.14 KiB)

شكل(58):صورة اصلية مع ثلاثة نماذج للضغط بطريقة فقدان بيانات

توجد لدينا عدة طرق للنوعين :-

- بدون فقدان بيانات توجد عدة طرق لها يمكن أن نأخذ بعضها كالأتي:

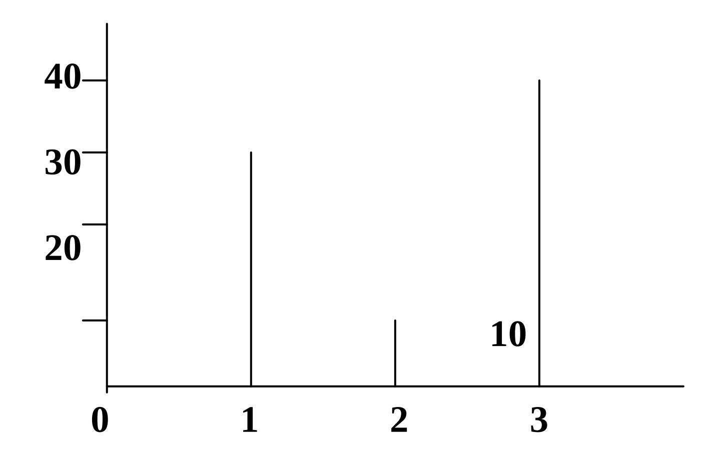
أ- طريقة هوفمان HUFFMAN CODING

هي الطريقة التي طورت من قبل العالم هوفمان سنة 1952 معتمدة على إيجاد أقل قيمة لطول الشفرة لذا نستخدم هنا قانونين هما :-

1. **قانون Entropy**
2. **قانون الطول Length**

الخوارزمية لهذه الطريقة هي نستخدم بطورين
الطور التقدمي والطور التراجعي

مثال // لدينا جزء من الصورة التالية 2 bit / Pixel أي تساوي 4 مستويات رمادية ... المطلوب تطبيق هذه المستويات من خلال الهيستوغرام التالي إلى طريقة هوفمان علما أن الRow يزداد 10 في كل مره



الحل //

1- إيجاد قيم الGray Level

$$g_o = 20$$
$$g_1 = 30$$
$$g_2 = 10$$
$$g_3 = 40$$

نلاحظ أن القيم المستخدمة للgray أكبر بكثير من قيمة الواحد (أقصى احتماليه تساوي واحد) لذا يجب تحويل القيم إلى قيم صغيرة لذا سوف نقوم بجمع الأعداد ونقسم كل عدد على مجموعهما بحيث تكون مجموع الأعداد المستخلصة هي واحد ...

$$20 + 30 + 10 + 40 = 100$$
$$g_o = 20/100 = 0.2$$
$$g_1 = 30/100 = 0.3$$
$$g_2 = 10/100 = 0.1$$

$g_3 = 40/100 = 0.4$

$0.2 + 0.3 + 0.1 + 0.4 = 1$

2- نرتب الأعداد التي حصلنا عليها تنازليا وكالتالي :-

$g_3 \rightarrow 0.4$
$g_1 \rightarrow 0.3$
$g_0 \rightarrow 0.2$
$g_2 \rightarrow 0.1$

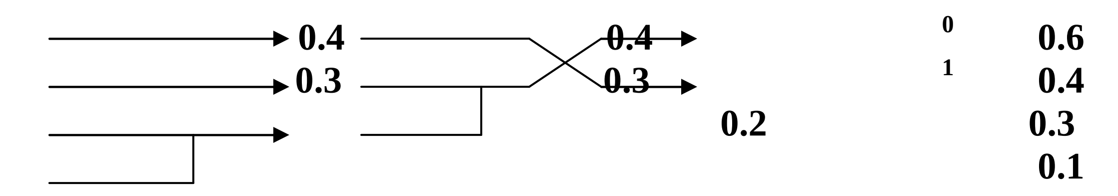
3- نأخذ أقل قيمتين وهما 0.1 و 0.2 ونجمعهم فتصبح 0.3 ونرتب النتيجة تنازليا أيضا وبعدها نأخذ أقل قيمتين ونجمعهما وهكذا إلى أن تبقى لدينا قيمتين فقط وكتالي :-

-4

$g_3 \rightarrow$ 0.4		0.4	$\rightarrow$ 0.4
$g_1 \rightarrow$ 0.3		0.3	$\rightarrow$ 0.3
$g_0 \rightarrow$ 0.2		0.2	$\rightarrow$ 0.3
$g_2 \rightarrow$ 0.1		0.1	

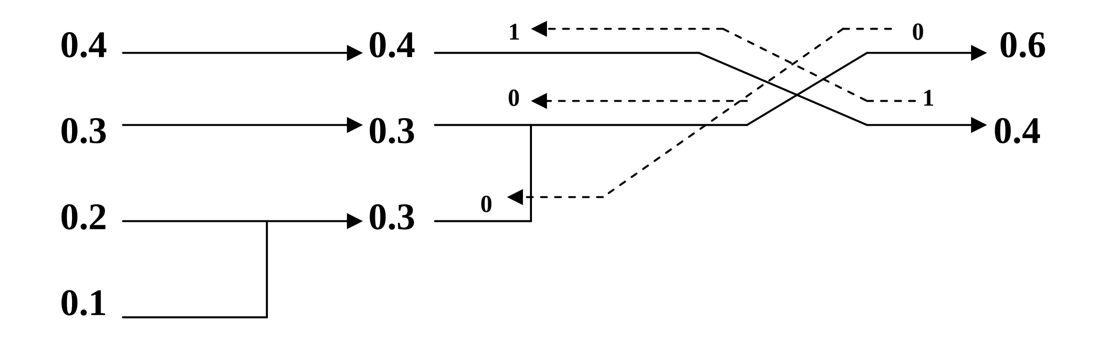


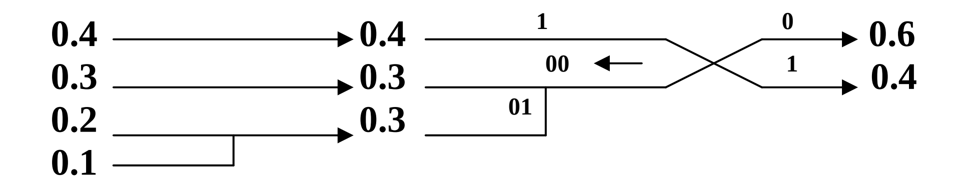
بعد أن أكملنا الطور التقدمي نأتي الآن إلى الطور التراجعي (نستخدم الشفرة)
الأولى 0.6 تأخذ الشفرة 0
والثانية 0.4 تأخذ الشفرة 1

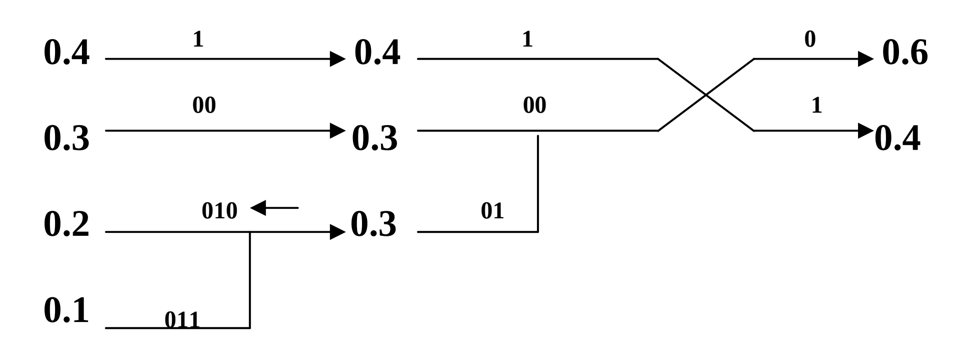


أما تفرعاتها أو أولادها فتأخذ الأولى 0 والثانية تأخذ 1 بالإضافة إلى قيمتها وهكذا .

.







فنحول الأرقام إلى Code

Original gray level (natural code)	Probability	Huffman code
g_0 : 00_2	0.2	010_2
g_1 : 01_2	0.3	00_2
g_2 : 10_2	0.2	011_2
g_3 : 112	0.4	1_2

1- قانون Entropy

Entropy = - $^{3}\sum_{i=0}$ Pi Log$_2$ (Pi) ………….(50)

Entropy = - $^{3}\sum_{i=0}$ Pi Log$_2$ (Pi)
= - [(0.2) Log$_2$ (0.2) + (0.3) Log$_2$ (0.3) + (0.1) Log$_2$ (0.1) + (0.4) Log$_2$ (0.4) ≈ 1.846 bits /pixel

2- قانون الطول Length

$$\textbf{Length} = \sum_{i=0}^{L-1} L_i P \quad ....(51)$$

$$= 3(0.2) + 2(0.3) + 3(0.1) + 1(0.4) = 1.9 \text{ bits / pixel}$$

ملاحظة//
يوجد قانون خاص بإيجاد $Log_2(x)$

$$\textbf{Log}_2 (x) = \textbf{Log}_{10} (x) * 3.322 \quad ....(52)$$

طريقه Huffman التي ابتكرها العالم David Huffman تعتمد علي إعطاء الحرف-أو كلمة (رمز) - كود خاص به بحيث لا يكون هناك تكرار في المعلومات اللازمة للتفرقة بين الحروف و بعضها البعض– كود ال - ASCIIبحيث يأخذ الحرف –أو الرمز- الأكثر تكرارا في الملف المراد ضغطه أقل كود ممكن مثل بت واحد أو 2 بت والحروف الأقل تكرارا تأخذ كود أطول.

أي أن طول الكود الخاص بكل رمز هو طول متغير Variable و ليس طول ثابت (8) Fixed بت كما كان الوضع في نظام ال ASCII و لكن يجب أن يظل من الممكن التفرقة بين كود كل حرف عند الحاجة لقراءة الملف المضغوط أو عند عملية فك الضغط ، ويتم استخدام شجرة ثنائيه Binary Treeمن أجل توليد هذه الأكواد للحروف -أو الرموز

خوارزمية بطريقة Huffman

يمكن تلخيص خطوات كما يلي :

1-إيجاد عدد مرات تكرار كل حرف في الملف النصي.

2-يتم تكوين قائمة من العناصر كل عنصر يحتوي علي الرمز و عدد مرات تكراره وهذه العناصر ستكون الأوراق - Leafs - للشجرة الثنائية.

3-اختر العنصرين من القائمة الذين لديهم أقل عدد مرات تكرار و اجمع أرقام التكرار لكل منهم لتحصل علي عنصر جديد يحتوي علي المجموع و يكون الابن -child– الأيمن لهذا العنصر الجديد هو العنصر الأقل تكرارا في القائمة و الابن الأيسر له هو العنصر الأقل الذي يليله ، ثم احذف العنصران اللذان تم اختيارهما من القائمة و أضف العنصر الجديد في القائمة بترتيبه.

4-يتم تكرار الخطوة رقم 3 لحين الحصول علي عنصر واحد في القائمة هذا العنصر سيكون ال rootللشجرة الثنائية التي سيتم توليد الأكواد بواسطتها.

5-نقوم بزيارة كل leaf (العنصر الذي ليس له أبناء في الشجرة) بداية من ال root, بحيث إذا انعطفنا يمينا يتم إضافة (0) للكود و إذا انعطفنا يسارا نضيف (1) للكود الخاص بالحرف الموجود في ال leaf التي سنزورها والكود الضي سينتج من الأصفار والوحياد التي كوناها عبر المسار من ال rootإلي ال Leaf سيكون هو كود الحرف الموجود في ال leafالتي تم زيارتها.

- العملية العكسية (فك الضغط):

الحرف و عدد مرات تكراره- وهذه المعلومات يتم قراءتها عند بداية قراءة الملف المضغوط ثم يتم تكوين الشجرة الثنائية مرة أخري داخل الذاكرة كما سبق تماما .

ثم نبدأ بقراءة الملف المضغوط بالبت Bit by Bit و نقف عند root الشجرة الثنائية، فإذا ما كانت البت المقروءة "1" ننتقل لليسار أو "0" فننتقل لليمين داخل الشجرة، ثم نختبر ما إذا كان العنصر الذي نقف عليه الآن داخل الشجرة

- فإذا كان Leaf نقرأ الحرف الموجود بداخله مثلا S و نكتبه في الملف الخرج (فك الضغط) ، ثم نعود مرة أخري إلي ال rootو نتابع القراءة من الملف المضغوط .

- أما إذا لم يكن Leaf نتابع قراءة البت التالية من الملف المضغوط وننتقل داخل الشجرة مرة أخري علي حسب قيمة البت المقروءة ثم نختبر ما إذا كانت Leaf أم لا

وهكذا حتى ننتهي من قراءة كل ال Bitsالموجودة داخل الملف المضغوط .

مثال//استخدم طريقة Huffman Encodingلسلسلة التالية :

AAAABCDEEEFFGGGH

الحل//

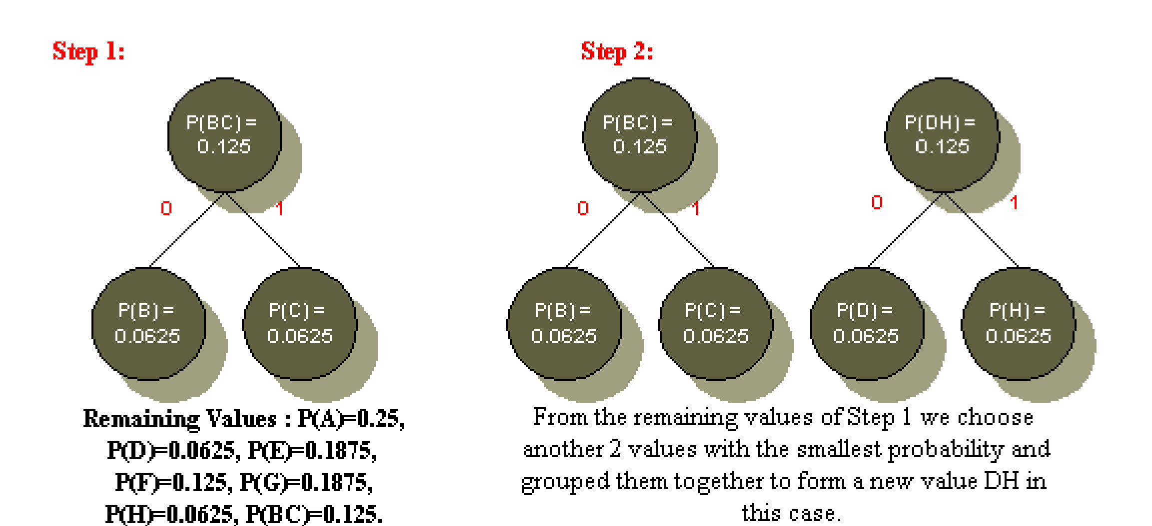
* نجد تكرارات كل عنصر

A : 4, B : 1, C : 1, D : 1, E : 3, F : 2, G : 3, H :1

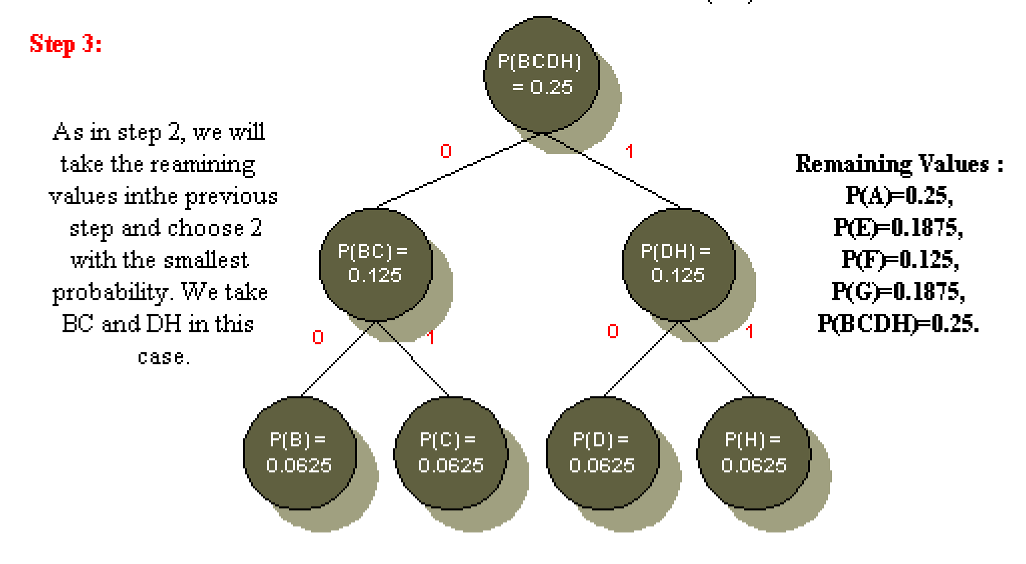
Based on the frequency count the encoder can generate a statistical model reflecting the probability that each value will appear in the data stream :

*نحول القيم كالتالي:

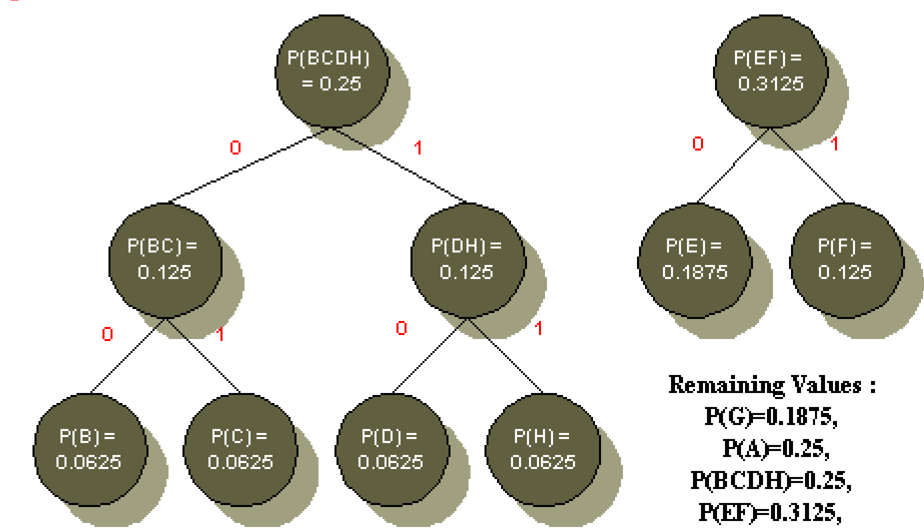
A : 0.25, B : 0.0625, C : 0.0625, D : 0.0625, E : 0.1875, F : 0.125, G : 0.1875, H : 0.0625



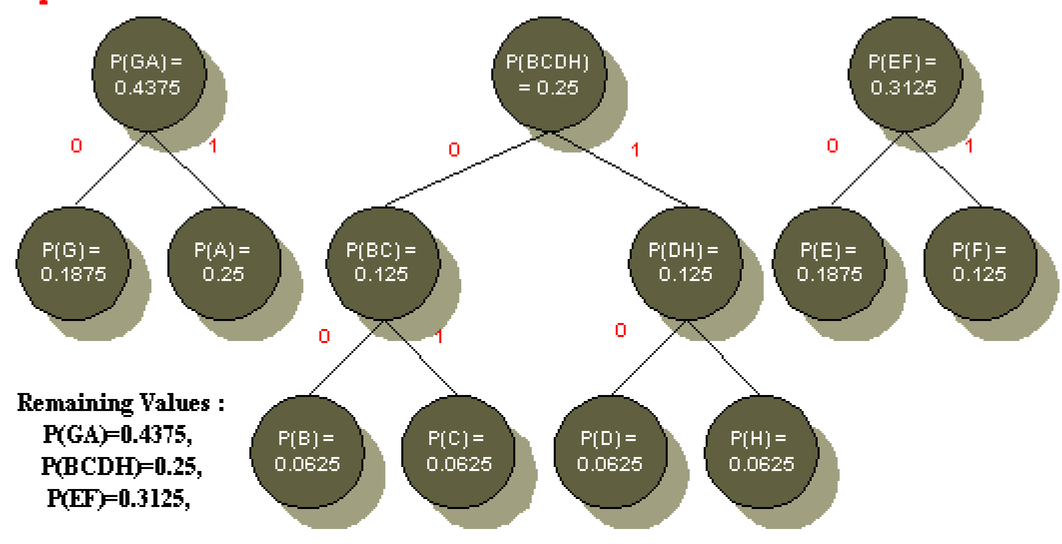
Remaining Values : P(A)=0.25, P(E)=0.1875, P(F)=0.125, P(G)=0.1875, P(BC)=0.125, P(DH)=0.125.



Step 4:


P(BCDH) = 0.25
0
1
P(BC) = 0.125
P(DH) = 0.125
0
1
0
1
P(B) = 0.0625
P(C) = 0.0625
P(D) = 0.0625
P(H) = 0.0625
P(EF) = 0.3125
0
1
P(E) = 0.1875
P(F) = 0.125
Remaining Values :
P(G)=0.1875,
P(A)=0.25,
P(BCDH)=0.25,
P(EF)=0.3125,

Step 5:


P(GA) = 0.4375
0
1
P(G) = 0.1875
P(A) = 0.25
P(BCDH) = 0.25
0
1
P(BC) = 0.125
P(DH) = 0.125
0
1
0
P(B) = 0.0625
P(C) = 0.0625
P(D) = 0.0625
P(H) = 0.0625
P(EF) = 0.3125
0
1
P(E) = 0.1875
P(F) = 0.125
Remaining Values :
P(GA)=0.4375,
P(BCDH)=0.25,
P(EF)=0.3125,



Step 6:

P(BCDHEF) = 0.5625

P(GA) = 0.4375 | P(BCDH) = 0.25 | P(EF) = 0.3125

P(G) = 0.1875 | P(A) = 0.25 | P(BC) = 0.125 | P(DH) = 0.125 | P(E) = 0.1875 | P(F) = 0.125

P(B) = 0.0625 | P(C) = 0.0625 | P(D) = 0.0625 | P(H) = 0.0625

Remaining Values :
P(GA)=0.4375,
P(BCDHEF)=0.5625.

Step 7:

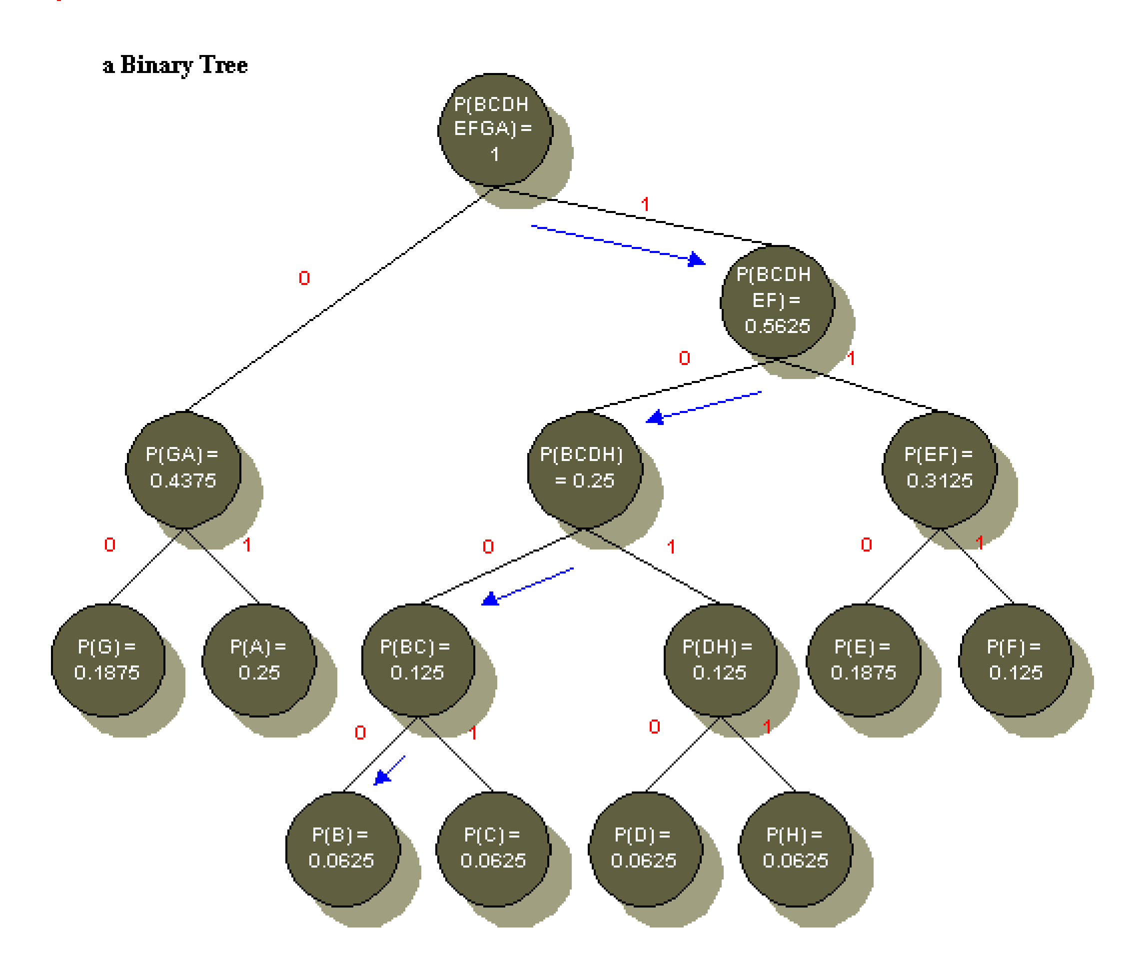
P(BCDHEFGA) = 1

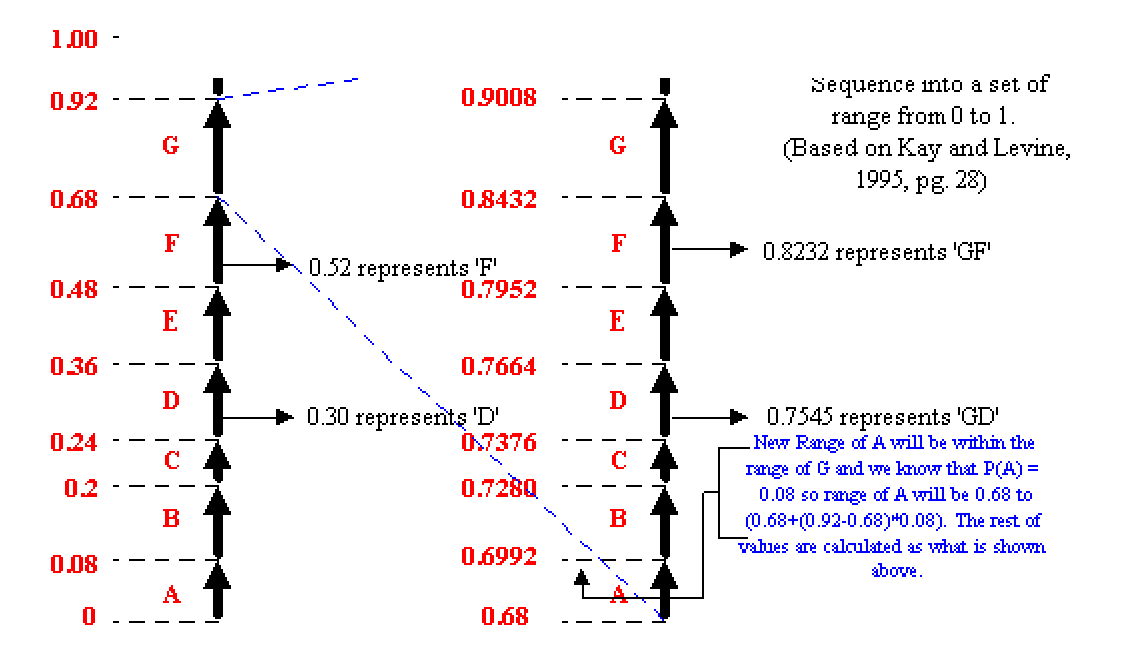
P(BCDHEF) = 0.5625

P(GA) = 0.4375 | P(BCDH) = 0.25 | P(EF) = 0.3125

Finally ...

a Binary Tree





. عيوبها و**Static** Huffman

أنه لابد من أن يكون لدي كل من الطرفين(الطرف الذي يقوم بضغط الملف و الطرف الذي يقوم بفك الضغط) معلومات عن ال Huffman Tree و هذه تعتبر overhead علي حجم الملف المضغوط .

العيب الثاني يتمثل في أنه لابد من قراءة الملف المراد ضغطه مرتين ، مرة لإيجاد تكرار كل حرف داخل النص و المرة الثانية عند إنشاء الملف المضغوط بعد توليد الأكواد للحروف ، وفي حالة الملفات الكبيرة الحجم هذا يعد مشكلة لأن العملية ستسغرق وقت طويل .

و في تطبيق آخر لطريقة Huffman مثل إرسال البيانات عبر الشبكات، نجد أن الطرف المستقبل لابد أن ينظر حتى ينتهي المرسل من عمله كاملا ثم يرسل له معلومات الشجرة الثنائية و البيانات المضغوطة ، وهذا يعد إهدار للوقت و المصادر- Resources -لأن أحد الطرفين يظل بلا عمل) (Idleحتى ينتهي الآخر من عمله .

ب- تشفير بطريقة(Run Length coding (RLC .

تعتمد على حساب عدد العناصر المتجاورة التي لها نفس القيمة اللونية ويسمى هذا العدد ب Run Length.

توجد العديد من الطرق لهذة الطريقة اهمها استخدامها مع الصور الثنائية التي طورت فيما بعد لاستخدامها مع الصور ذات التدرجات الرمادية والصور الملونة.

مثال// لديك جزء الصورة التالي المطلوب استخدام (Run Length **coding (RLC** لضغطها؟

0	0	0	0
0	1	1	0
1	1	1	0
0	0	1	1

الحل//
R1=4
R2=1,2,1
R3=0,3,1
R4=2,2

حيث أن العدد الاول يمثل عدد الاطفار بالصف والعدد الثاني يمثل الواحدات

مثال//
36 0 0 0 12
فإنها تكتب:

3 – 12 و يعني الرقم الأول أنه يجب أن نتخطى 3 بتأت (أي لا يتم حفظها) و أن الرقم القادم يكون -12

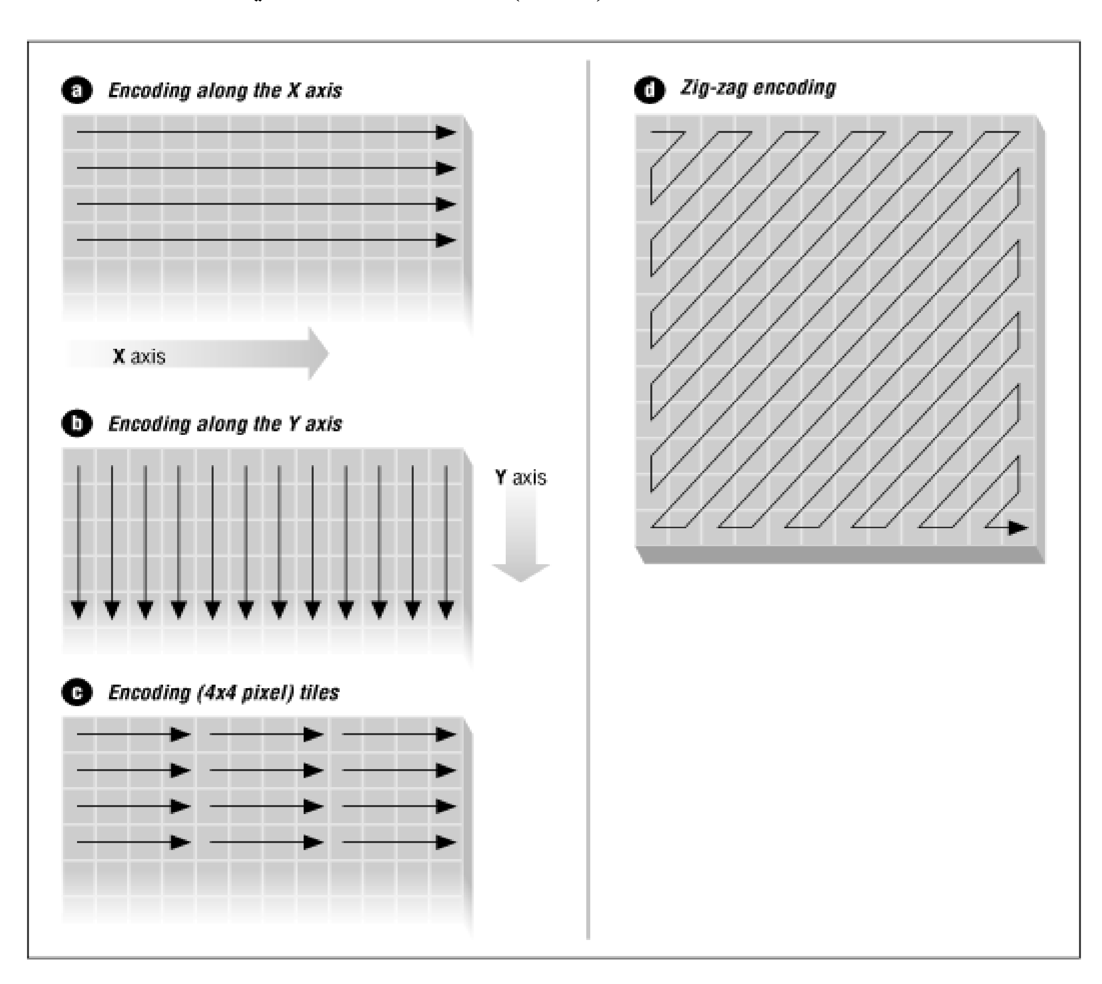
مثال//اذا كان لدينا السلسلة التالية

AAAAAAAAAAAAAAA

The same string after RLE encoding would require only two bytes:

15A

نلاحظ أن اطوال التكرارات لكل رمز غير ثابتة(متغيرة) يوضحها الشكل التالي:

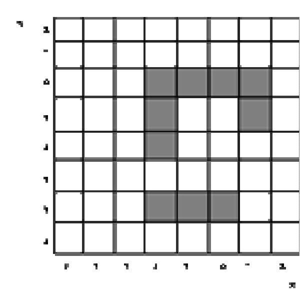


شكل(59):اطوال طريقة تشفير طول التنفيذ

والمخطط الانسيابي يوضح الخطوات:

شكل(60):مخطط انسيابي **Basic run-length encoding**

مثال//استخدم Run length coding **للشكل ادناة:**



شكل(61) صورة بطريقة Run length coding

الحل//

{ (2,4,6) , (4,4,4), (5,4,4,7,7) , (6,4,7) }

{ (8) , (3,3,2) , (8), (3,1,4) , (3,1,2,1,1), (3,4,1) , (8) }

ملاحظة//
قانون Run Length coding (RLC)
المستخدم في الصورة الرمادية يتمثل بتعريف الزوج(G,L) حيث أن

G:القيمة اللونية
L(run):عدد تكرارات القيمة اللونية

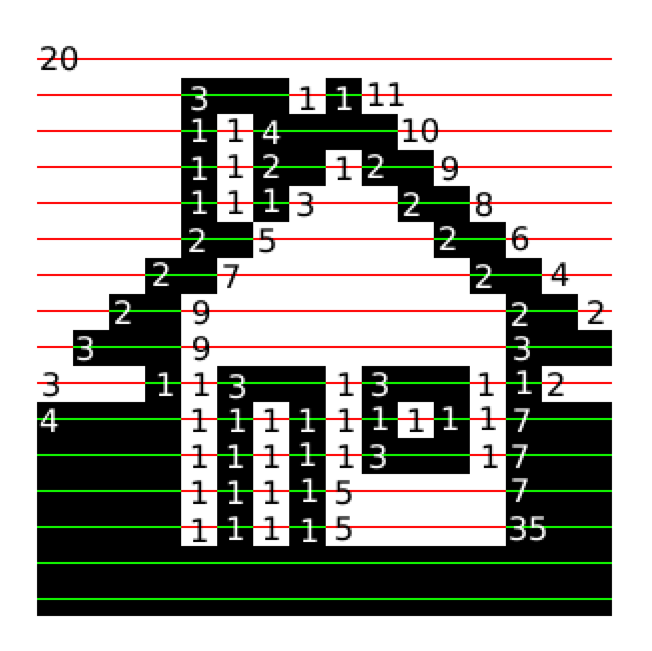
مثال// اذا كانت لدينا صورة ابعادها 8*8 حيث اربعة لكل بكسل احسب **coding (RLC)** Run Length لها:

10	10	10	10	10	10	10	10
10	10	10	10	10	14	14	14
10	10	10	10	10	10	6	6
0	0	0	10	10	10	0	0
5	5	0	0	0	0	0	0
5	5	5	10	10	9	9	10
5	5	5	4	4	4	0	0
0	0	0	0	0	0	0	0

الحل//
R1=10,8
R2=10,5,14,3
R3=10,6,6,2
R4=0,3,10,3,0,2
R5=5,2,0,6
R6=5,3,10,2,92,10,1
R7=5,3,4,3,0,2
R8=0,8

قانون (**RLC** Run Length **coding)** =
{0,8,10,810,5,14,310,6,6,20,3,10,3,0,25,2,0,65,3,10,2,92,10,15,3,4,3,0,2}

مثال// اوجد Run-length encoding للصورة التالية؟



شكل(62):تحديد Run-length encoding صورة بيت

70,
5, 25,
5, 27,
4, 26,
4, 25,
6, 24,
6, 23,
3, 2, 3, 22,
3, 2, 3, 21,
3, 5, 2, 20,
3, 5, 2, 19,
3, 7, 2, 18,
3, 7, 2, 17,
14, 16,
14, 15,
3, 11, 2, 14,
3, 11, 2, 13,
3, 13, 2, 12,
3, 13, 2, 11,
3, 15, 2, 10,
3, 15, 2, 8,
6, 12, 6, 6,
6, 12, 6, 64

تم تحديد المربعات البيضاء والسوداء الموجوة في صورة البيت اعلاة وقد تم تقليل عدد القيم إلى 72 باختيار اقصى طول 15 وضع القيمة صفر اذا تجاوز الحد حسب القيم التالية:

```
70,                        15,  0, 15,  0, 15,  0,
10,

 5, 25,                     5, 15,  0, 10,

 5, 27,                     6, 15,  0, 12,

 4, 26,                     4, 15,  0, 11,

 4, 25,                     4, 15,  0, 10,

 6, 24,                     6, 15,  0,  9,

 6, 23,                     6, 15,  0,  8,

 3,  2, 3, 22,              3,  2,  3, 15,  0,  7,

 3,  2, 3, 21,              3,  2,  3, 15,  0,  6,

 3,  5, 2, 20,              3,  5,  2, 15,  0,  5,

 3,  5, 2, 19,              3,  5,  2, 15,  0,  4,

 3,  7, 2, 18,              3,  7,  2, 15,  0,  3,

 3,  7, 2, 17,              3,  7,  2, 15,  0,  2

14, 16,                    14, 15,  0,  1

14, 15,                    14, 15,

 3, 11, 2, 14,              3, 11,  2, 14,

 3, 11, 2, 13,              3, 11,  2, 13,

 3, 13, 2, 12,              3, 13,  2, 12,

 3, 13, 2, 11,              3, 13,  2, 11,

 3, 15, 2, 10,              3, 15,  2, 10,
```

3, 15, 2, 8, 6, 12, 6, 6, 6, 12, 6, 64 15, 0, 15, 0, 4	3, 15, 2, 8, 6, 12, 6, 6, 6, 12, 6, 15, 0, 15, 0,

يوجد 113*nibbles* وحدة (تتمثل 4—0) حيث نحتاج إلى 57 بايت لخن كل القيم التي تكون اقل أو تساوي 93 بايت ونحتاج إلى 750 اذا استخدمنا بايت لكل بكسل.

ج-طريقة Bit-Plane RLC:

تعتبر طريقة موسعة لطريقة **RLC** للصور المستوى الرمادي لكل سطح خاص بالبت الواحد مستخدمين الشفرة الطبيعية والشفرة الرمادية محددين بقيمتي الصفر والواحد .
أن نسبة الضغط بهذة الطريقة يتمثل من 0.5 إلى 1.2 مستخدمين 8 بت للصور الاحادية اللون يمكن توضيح الطريقة بالمثال ادناة

مثال//الشكل ادناة يبين شفرة الطبيعية والشفرة الرمادية لاربعة بتأت مستخدما القيم الثنائية



شكل(63): شفرة الطبيعية والشفرة الرمادية

طريقة ايجاد الشفرة الرمادية:

1-نأخذ الرقم الاول من النظام الثني المراد تحويلة اذا كان (1) ننزل (1)
2- اذا كان الرقم (0) ننزل (0)
3- نقارن الرقم مع الرقم الي بعدة وللكل اذا كان الرقمان متشابهان نضع (0)والا ننزل (1)

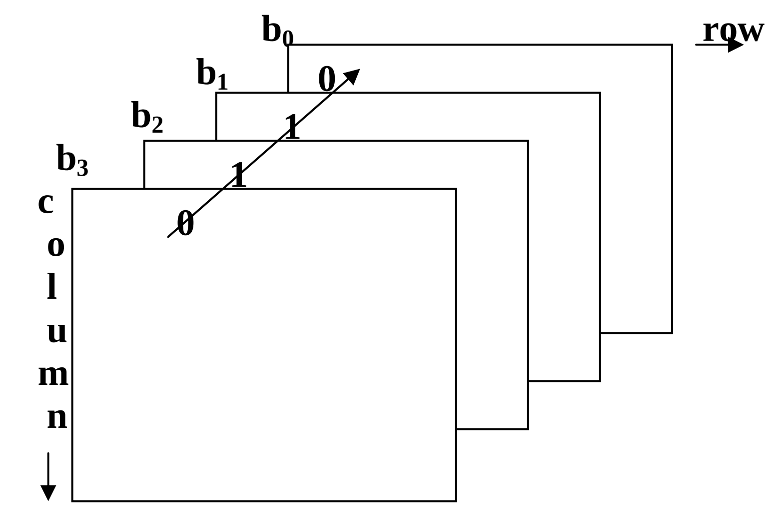
مثال//حول 00110011 إلى شفرة الرمادية
الحل//
00101010

مثال//حول 11001001101 إلى الشفرة الرمادية

الحل//10101101011

a:bits/pixel designation

b_3	b_2	b_1	b_0
0	0	0	0
0	0	0	1
0	0	1	0
⋮	⋮	⋮	⋮
0	1	1	0
1	1	1	1



شكل(64):طريقة Bit – plane Run – length

ملاحظة//يمكن أن نفترض أن قيمة طريقة أخرى و هي باخذ كل قيمة مفردة للطول ومعاملين رمزين للتمثيل هما(G,L) حيث G تمثل المستوى الرمادي ، L تمثل الطول تكون هذة الطريقة فعالة اذا كانت عدد المستويات الرمادية قليل.

Decimal	4 – bit natural code	4 – bit gray code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

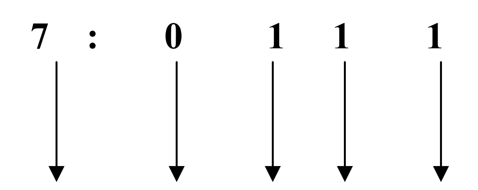
7 : 0 1 1 1

↓ ↓ ↓ ↓ ↓

8 : 1 0 0 0

b. the natural code transition of 7 to 8 changes all four bits

شكل(65): الشفرة الطبيعية مقابل الشفرة الاعتيادية



شكل(66):الشفرة الطبيعية المستخدمة للتحويل

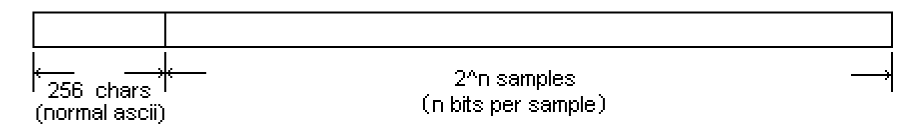
د- طريقة(Lempel-Ziv-Welch (lzw:

و تم تطوير هذا النوع من تقنية الضغط في السبعينات على يد ثلاثة أساتذة و هم:
Abraham Lempel و Jacob Zivو تم تعديلة و تجديدة على يد الأستاذ Welch Terry و تقديرا لهم ،سميت التقنية بأسمائهم
LZW = Lempel-Ziv-Welch

مقدار الضغط يعتمد على درجة تغيّر اللون في كل سطر من البكسلات، عملية الضغط تقوم على الآلية التالية:
إذا كان هناك بكسلين أو أكثر في السطر الواحد يحملان نفس اللون، فإن هذه البكسلات تسجّل كوحدة منفردة.
من هنا نستنتج أن الصورة الحاوية على شرائح لونية أفقية سوف تكون مضغوطة أكثر بكثير من الصورة الحاوية على شرائح عمودية، لأن كل خطّ أفقي سيتم تخزينه كوحدة واحدة.

الصور التي تحتوي على مناطق كبيرة ذات لون متجانس مثل السماء، الثلج، الغيوم وغيرها تكون مضغوطة أكثر من الصور الحاوية على الكثير من التدرجات اللونية. عند حفظ الصورة ذات العمق اللوني 24 بت على هيئة GIF ينبغي أولاً تخفيض عمق اللون إلى مستوى 8 بت.

معظم التطبيقات تسمح بذلك وبسهولة ،عند العمل مع صور التدرجات الرمادية Grayscale (الأبيض والأسود) فأن هيئة GIF تصلح للعمل بصورة جيدة، ذلك أن معظم التطبيقات تستخدم عمق لوني 8 بت (256 تدرج رمادي) مع صور الأبيض والأسود.



شكل(67):الهيكل طريقةLempel-Ziv-Welch (lzw)

خوارزمية LZW:

Compression

The LZW Compression Algorithm can be summarized as follows:

```
w = NIL;
while ( read a character k )
{
   if wk exists in the dictionary
      w = wk;
   else
   {
      add wk to the dictionary;
      output the code for w;
      w = k;
   }
}
```

تمتلك الخوارزمية قاموس متكون من 4094من المدخلات اول 256 منها تمثل شفرات الاسكي .

ASCII

مثال//ادخال الخيط الرمزي."^WED^WE^WEE^WEB^WET"

w	k	output	index	symbol
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE

E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

خوارزمية فك التشفير:.

Decompression

The LZW Decompression Algorithm is as follows:

```
read a character k;
output k;
w = k;
while ( read a character k )
// k could be a character or a code.
{
    entry = dictionary entry for k;
    output entry;
    add w + entry[0] to dictionary;
    w = entry;
}
```

Decompression Example

Input string is "^WED<256>E<260><261><257>B<260>T".

w	k	output	index	symbol
	^	^		
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
<256>	E	E	260	^WE
E	<260>	^WE	261	E^
<260>	<261>	E^	262	^WEE

<261>	<257>	WE	263	E^W
<257>	B	B	264	WEB
B	<260>	^WE	265	B^
<260>	T	T	266	^WET

مثال//استعمل طريقة lzw لضغط النص التالي

itty bitty bit bin

الحل//

نقوم بايجاد القيم التي تمثل كل حرف بنظام الاسكي

32	space
98	b
105	i
110	n
116	t
121	y

: LZW Example 1 Dictionary

نقوم بتطبيق الطريقة على القيم المستحصلة

Encoding → Transmission → Decoding

Input		New dictionary entry		9-bit characters transmitted		New dictionary entry		Output
105	i	–	–	256	(start)	–	–	–
116	t	258	it	105	i	–	–	i
116	t	259	tt	116	t	258	it	t
121	y	260	ty	116	t	259	tt	t
32	space	261	y-space	121	y	260	ty	y
98	b	262	space-b	32	space	261	y-space	space
105	i	263	bi	98	b	262	space-b	b
116	t	–	–	–	–	–	–	–
116	t	264	itt	258	it	263	bi	it
121	y	–	–	–	–	–	–	–
32	space	265	ty-space	260	ty	264	itt	ty
98	b		–	–	–	–	–	–
105	i	266	space-bi	262	space-b	265	ty-space	space-b
116	t	–	–	–		–	–	–
32	space	267	it-space	258	it	266	space-bi	it
98	b	–	–	–		–	–	–
105	i	–	–	–		–	–	–
110	n	268	space-bin	266	space-bi	267	it-space	space-bi
–	–	–	–	110	n	268	space-bin	n
–	–	–	–	257	(stop)	–	–	–

مثال//طبق طريقة lzw للنص التالي

itty bitty nitty grrritty bit bin

الحل// نقوم بايجاد القيم التي تمثل كل حرف بنظام الاسكي

32	space	110	n
98	b	114	r
103	g	116	t
105	i	121	y
256	clear dictionary	257	end of transmission

Encoding →				Transmission →		Decoding		
Input		New dictionary entry				New dictionary entry		Output
105	i	–	–	256	(start)	–	–	–
116	t	258	it	105	i	–	–	i
116	t	259	tt	116	t	258	it	t
121	y	260	ty	116	t	259	tt	t
32	space	261	y-space	121	y	260	ty	y
98	b	262	space-b	32	space	261	y-space	space
105	i	263	bi	98	b	262	space-b	b
116	t	–	–	–	–	–	–	–
116	t	264	itt	258		263	bi	it
121	y	–	–	–	–	–	–	–
32	space	265	ty-space	260	ty	264	itt	ty
110	n	266	space-n	32	space	265	ty-space	space
105	i	267	ni	110	n	266	space-n	n
116	t	–	–	–	–	–	–	–
116	t	–	–	–	–	–	–	–
121	y	268	itty	264	itt	267	ni	itt
32	space	–	–	–	–	–	–	–
103	g	269	y-space-g	261	y-space	268	itty	y-space
114	r	270	gr	103	g	269	y-space-g	g
114	r	271	rr	114	r	270	gr	r
114	r	–	–	–	–	–	–	–
105	i	272	rri	271	rr	271	rr	rr
116	t	–	–	–	–	–	–	–
116	t	–	–	–	–	–	–	–
121	y	–	–	–	–	–	–	–
32	space	273	itty-space	268	itty	272	rri	itty
98	b	–	–	–	–	–	–	–
105	i	274	space-bi	262	space-b	273	itty-space	space-b
116	t	–	–	–	–	–	–	–
32	space	275	it-space	258	it	274	space-bi	it
98	b	–	–	–	–	–	–	–
105	i	–	–	–	–	–	–	–
110	n	276	space-bin	274	space-bi	275	it-space	space-bi
–	–	–	–	110	n	276	space-bin	n
–	–	–	–	257	(stop)	–	–	–

2- طرق الضغط الحاوية على فقدان البيانات : Lossy Compression Methods

للحصول على نسبة ضغط علية يتم استخدام هذا النوع من طرق الضغط والتي تتطلب موازنة بين نوعية الصورة الناتجة ونسبة الضغط وتستخدم هذه الطرق عادة مع الصورة المعقدة Complex Image تنفذ طرق الضغط الحاوية على فقدان البيانات في المجالين ألحيزي Spatial ومجال التحويل Transformation ومن هذه الطرق :-

1- طريقة قطع البلوكات Block Truncation Coding

وهي من الطرق التي تعمل في المجال ألحيزي حيث يتم تقسيم الصورة إلى مجموعة من الصور الجزئية الصغيرة تسمى Sub images وأحيانا تسمى بالبلوك . وتعتمد هذه الطريقة على تقليل المستويات اللونية في كل صورة جزئية أو Block .
تقطع الصورة إلى 4*4 بلوكات كل بلوكيحتوي 4 بايت (2 بايت لخزن مستويين و2 بايت لخزن خيط رمزي للبت من الصفر والواحد)كالأتي:

a. Divide image into 4*4 blocks

b. Find high and low values for blocks

1	1	0	1
0	0	0	0
1	1	1	1
.	1	0	1

High gray value
Low gray value

c. Assign a to each pixel less then the mean , 1 to each pixel greater than the mean

Row 1	**Row 2**
Row3	**Row 4**

d. Assign a to each pixel less then the mean , 1 to each pixel greater than the mean

شكل(68):طريقة طريقة قطع البلوكات Block Truncation Coding

2-ترميز التحويل Transformation coding:

تعمل هذه الطريقة في مجال التحويل حيث يتم إسقاط عناصر الصورة الأصلية إلى مجال رياضي أخر يسمى Transformation أن الهدف الأساسي من التحويلات هو لتقليل الارتباط بين عناصر الصورة وتجميع المعلومات في عدد قليل من معاملات التحويل يشبه ترميز التحويل طريقة ترميز Block ألا أنها تعمل في مجال التحويل حيث يتم تقسيم الصورة إلى بلوكات ثم يتم حساب التحويل المستخدم لكل Block ومن هذه التحويلات Discrete Cosines Transformation (DCT) ، Discrete Fourier Transformation (DFT)



شكل(69):صورة ضغطتDCTحيث النسبة 5.1
حجم البلوك=64

نظرية التحويل

يقوم التحويل بإسقاط بيانات الصورة إلى مجال رياضي أخر عن طريق معادلة التحويل حيث يتم تحويل بيانات الصورة من المجال ألحيزي إلى المجال الترددي أو مجال التحويل .
الشكل العام لمعادلة التحويل على افتراض أن حجم الصورة هو n*n يكون كتالي :-

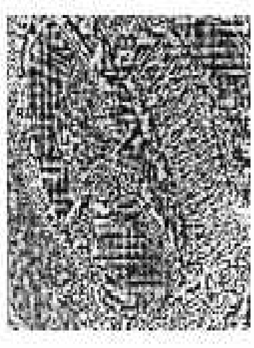
$$T(u, v) = \sum_{r=0}^{n-1} \sum_{c=0}^{n-1} I(r, c)\, B(r, c, u, v) \quad \ldots\ldots(53)$$

حيث u , v هي متغيرات مجال التحويل
I(r , c) هي بيانات الصورة
B(r ,c , u ,v) هي دالة التحويل

للحصول على بيانات الصورة من بيانات التحويل نطبق معادلة التحويل العكسي :-

$$I(r,c) = T^{-1}[T(u ,v)] = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u ,v)\, B^{-1}(r, c, u, v)....(54)$$





Example:
DCT compression with ratio 5.6
Left: Reconstructed image
Right: Difference image (right) with maximal difference of 125 greylevels

شكل(70):صورة ضغطتDCT
حجم البلوك=64

من الطرق الضغط بفقدان بيانات هي:

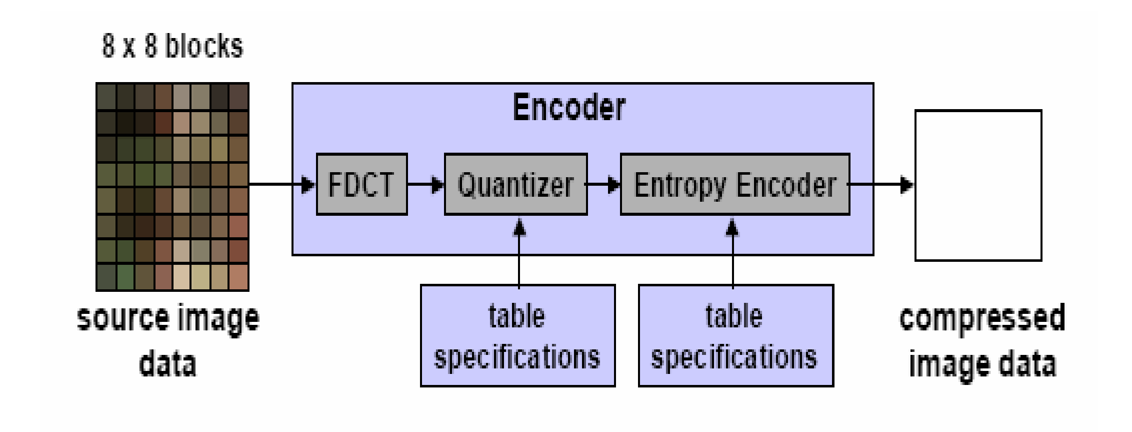
<u>طريقة JPEG</u> :

JPEG اختصار لكلمة Joint Photographic Expert Group، وهي الجهة القائمة على تطوير هذه الخوارزمية.

يعتبر JPEG في المعلوماتية طريقة معيارية شائعة لضغط الصور الرقمية مع خسارة في القيمة المعلوماتية للصورة. أشهر الإمتدادات المستخدمة لها هي .jpeg و .jfif و .jpg و .JPG و .JPE لكن .jpg يبقى أكثرها استخداما.

تعتبر - JPEG/JFIF - الاكثر استخداما لحفظ ونقل الصور الشمسية على شبكة الويب، إذ أنها مفضلة على امتدادات أخرى مثل GIF، الذي لا يسمح إلا ب256 لون مختلف، وهذا غير كاف للصور الشمسية، و PNG الذي ينتج ملفات كبيرة مقارنة ب JPEG/JFIF.

تقوم بضغط العالي الذي يضغط البيانات بنسبة 20 مرة تقريبا فمثلا إذا كانت تحتاج صورة 200 بت فإن استعمال خوارزمية الضغط هذه يمكن تقليصها إلى 10 بتأت.



شكل(71):خطوات طريقة JPEG

خوارزمية JPEG للضغط متناظرة أي أن الجهد اللازم للتشفير هو نفس الجهد اللازم لفك التشفير. وفي ما يلي شرح لكيفية عمل الخوارزمية في أبسط صيغاتها المسماة تشفير خط الأساس المتتالي Baseline Sequential Encoding والقائمة على الخطوات التالية:

***تغيير التشفير**

يتم تغيير التشفير من RGB (أحمر أخضر أزرق) إلى تشفير Y،U،V حيث Y هي الإضاءة و U وV هي التلون.

يتم التحويل بالطريقة التالية:

$$B0,144 + G0,587 + R0,299 = Y$$

$$128 + B0,5 + G0,331 - R0,169 - = U$$

$$128 + B0,081 - G0,419 - R0,5 = V$$

مع الإشارة إلى أن R،G،B،U،V،Y بين 0 و 255. كما أن المعاملات في التحويل تأتي من حساسية العين البشرية حيث أنها حساسة جدا في مجال اللون الأخضر في حين أنها أقل جساسية في مجال اللونين الأزرق و الأحمر ذلك فإن المعامل للون الأخضر هو الأكبر قيمة مقارنة بالمعاملين الأخرين.

***تقسيم الصورة إلى قطع 8x8 بكسل**

يتم تقسم الصورة إلى قطع متكونة من8 x 8 بكسل لتكون أسهل معالجة في الخطوة الموالية.

***تحويل جيب التمام**

يتم تحويل كل من المصفوفات 8x 8 عن طريق تحويل جيب التمام و نتحصل بذلك على مصفوفة تحتوي على التردد المكاني

*** إستعيان التردد المثالي**

في هذه الخطوة يمكن أن نفقد بعض البتات أو المعلومات من الصورة. حيث يتم إستعيان الترددات Frequency Samplingالبطيئة بطريقة جيدة(بطريقة تسمح برؤية الفوارق الصغيرة بين ترددين مختلفين) في حين يتم إستعيان الترددات السريعة بطريقة سيئة (أي أنه مثلا تردد 50 بكسل في وحدة طول و 60 بكسل في وحدة طول كلها تمثل ب 50 بكسل في وحدة طول أي لا يمكن التفريق بينهما. مع الإشارة إلى أنه يجب فهم كلمة بكسل في وحدة طول على أنها تردد مكاني حيث لا يمكن إتخاذ الهرتز كوحدة هنا لأن التردد ليس زمني).

و السبب في عمل هذا هو أن العين البشرية لا يمكنها التمييز بين الترددات المكانية السريعة فإن تأملت في خط طوله سنتمتر مثلا يتكون من 100 بكسل فإنك لن تلحظ فرقا بالنسبة لخط فيه 1000بكسل.

***مسح الصورة**

يتم بعد ذلك مسح الصورة بطريقة تجعل البيانات في شكل شعاع بدلا من مصفوفة و يتم المسح كما هو مبين في الصورة.

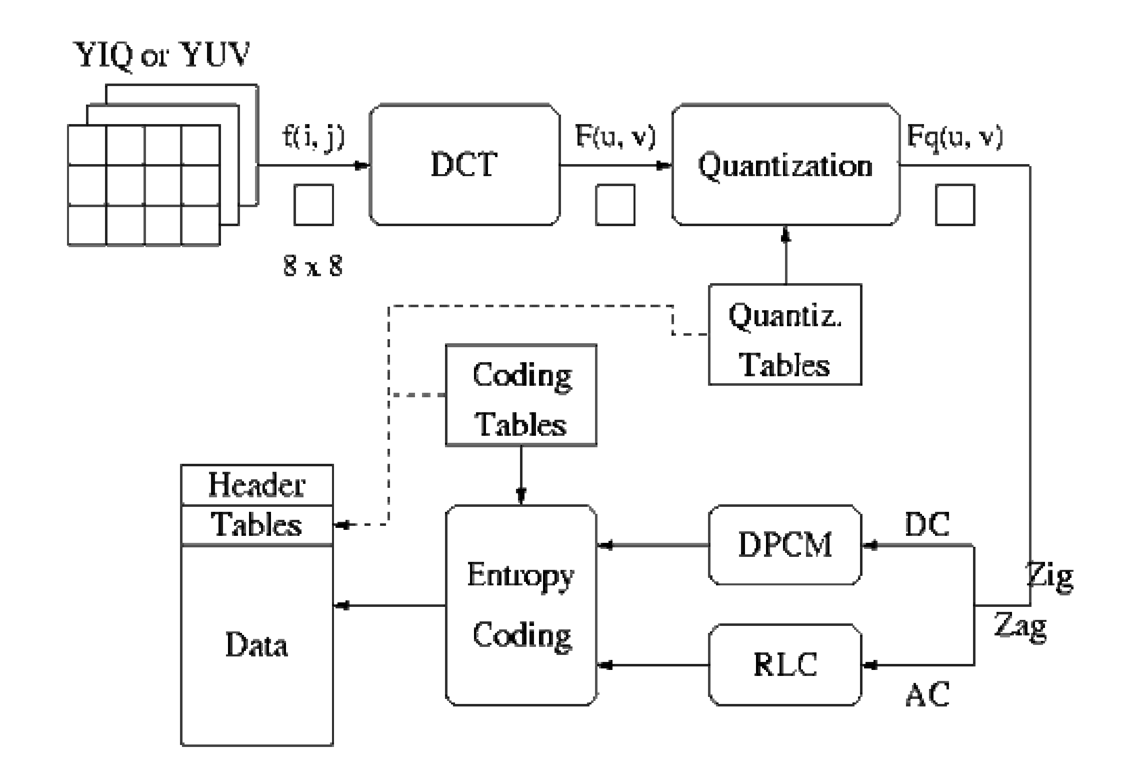
***تشفير بطريقة(Run Length coding (RLC** :

تعتمد على حساب عدد العناصر المتجاورة التي لها نفس القيمة اللونية ويسمى هذا العدد بـRun Length ، توجد العديد من الطرق لهذه الطريقة أهمها استخدامها مع الصور الثنائية التي طورت فيما بعد لاستخدامها مع الصور ذات التدرجات الرمادية والصور الملونة

*** تشفير بطريقة VLC**

VLC اختصار ل Variable Length Coding أو ما يعرف بتشفير هوفمان Huffman-Coding. حيث يتم حفظ كل عدد يظهر بالشعاع بطريقة أمثل من ناحية حجم البيانات ،حيث أن الأرقام التي تظهر بكثرة في الشعاع يتم تشفيرها بعدد صغير من البتات مثلا إذا كان الرقم 2 يظهر بكثرة في الشعاع فإنه يرمز له ب 0 عوض التشفير الثنائي العادي الذي يحتاج إلى بتان لحفظ الرقم 2 لأنه يحفظها 10.

أما الأرقام التي تظهر بقلة فإنه يتم تشفيرها بعدد كبير من البتات. يتم التشفير بطريقة حيث أنه لا يمكن أن يتشابه رقمان في بدايتهما يعني أنه لو كان أول بت تحصل عليه من الشعاع هو 0 فلا داعي لانتظار ما سيأتي بعده من بتأت لمعرفة معنى هذا البت و يمكن القول مباشرة أن 0 هي التشفير المقابل لرقم 5 مثلا. و تمثل الخطوتان الأخيرتان ما يعرف ب تشفير إنتروبي و هي طريقة تشفير لا تضيع معها معلومات.



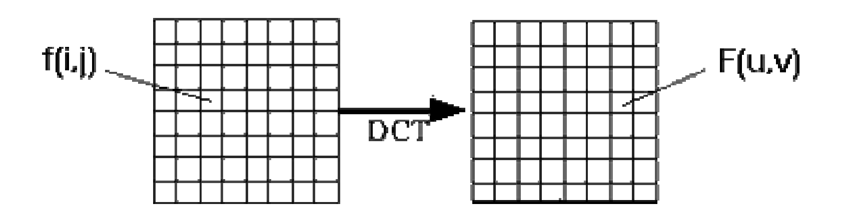
شكل(72):خطوات طريقةJPEG

الخطوات الاساسية:

- DCT (Discrete Cosine Transformation)
- Quantization
- Zigzag Scan
- DPCM on DC component
- RLE on AC Components
- Entropy Coding

1-تحويل جيب المتقطع(Discrete Cosine Transform (DCT .

يمكن تمثيل الخطوة الاولى بالشكل التالي:



شكل(73): Discrete Cosine Transform (DCT)

- **Discrete Cosine Transform (DCT):**

$$F(u,v) = \frac{\Lambda(u)\Lambda(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot f(i,j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

………………….(55)

Inverse Discrete Cosine Transform (IDCT):

$$\tilde{f}(i,j) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\Lambda(u)\Lambda(v)\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot F(u,v)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

……………….(56)

2- التعميم Quantization تستخدم لتقليل عدد البتات في العينة الواحدة

- F'[u, v] = round (F[u, v] / q[u, v])………………(57)

  Example: 101101 = 45 (6 bits).
  q[u, v] = 4 --> Truncate to 4 bits: 1011 = 11.

.يوجد نوعين منه:

1 -المنتظم Uniform Quantization

- Each F[u,v] is divided by the same constant *N*.

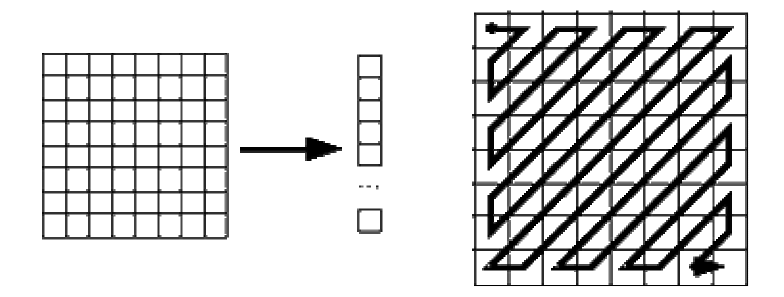
2 -غير المنتظم Non-uniform Quantization – Quantization Tables

- Eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner)
- The *Luminance Quantization Table* q(u, v) The *Chrominance Quantization Table* q(u, v)
- --

16	**11**	**10**	**16**	**24**	**40**	**51**	**61**	**17**	**18**	**24**	**47**	**99**	**99**	**99**	**99**
12	**12**	**14**	**19**	**26**	**58**	**60**	**55**	**18**	**21**	**26**	**66**	**99**	**99**	**99**	**99**
14	**13**	**16**	**24**	**40**	**57**	**69**	**56**	**24**	**26**	**56**	**99**	**99**	**99**	**99**	**99**
14	**17**	**22**	**29**	**51**	**87**	**80**	**62**	**47**	**66**	**99**	**99**	**99**	**99**	**99**	**99**
18	**22**	**37**	**56**	**68**	**109**	**103**	**77**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**
24	**35**	**55**	**64**	**81**	**104**	**113**	**92**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**
49	**64**	**78**	**87**	**103**	**121**	**120**	**101**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**
72	**92**	**95**	**98**	**112**	**100**	**103**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**

3- مسح Zig-zag Scan الذي يبين التردد الواطي لمجموعة المعاملات في اعلى المتجة

- Maps 8 x 8 to a 1 x 64 vector



Zig-zag Scan:(74)شكل

Differential Pulse Code Modulation (DPCM) on DC component-4

- DC component is large and varied, but often close to previous value.
- Encode the difference from previous 8 x 8 blocks -- DPCM

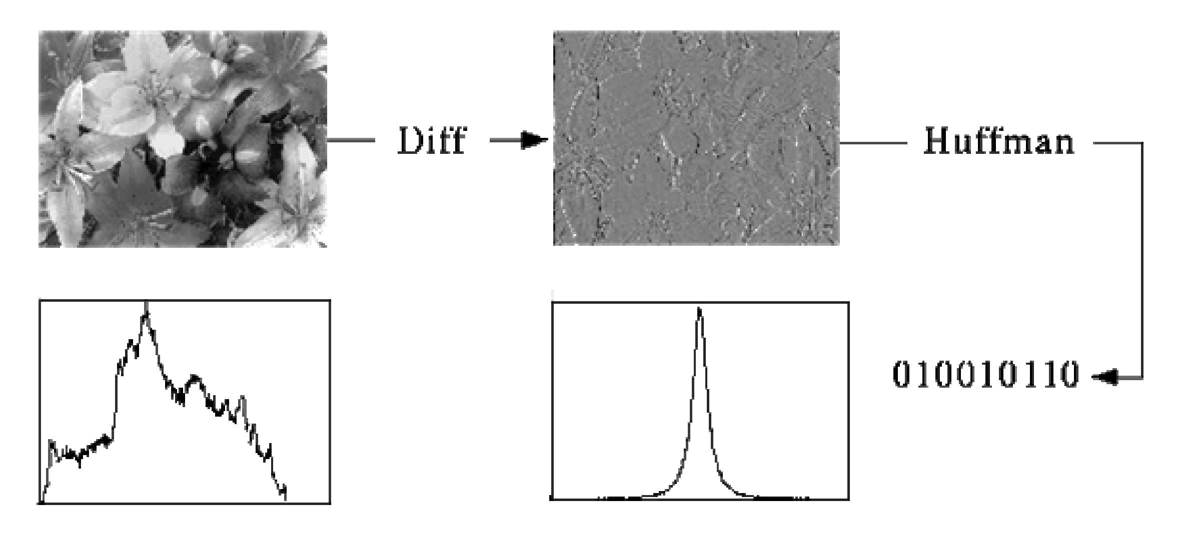
. Run Length Encode (RLE) on AC components-5

- 1 x 64 vector has lots of zeros in it
- Keeps *skip* and *value*, where *skip* is the number of zeros and *value* is the next non-zero component.
- Send (0,0) as end-of-block sentinel value.

Entropy Coding-6

- Categorize DC values into SIZE (number of bits needed to represent) and actual bits.

```
•     ------------------------------------
•        SIZE                  Value
•     ------------------------------------
•          1                    -1, 1
•          2                -3, -2, 2, 3
•          3                -7..-4, 4..7
•        4              -15..-8, 8..15
•        .                   .
•        .                   .
•        .                   .
•       10         -1023..-512, 512..1023
•     ------------------------------------
```



شكل(75): كيفية الضغط بطريقة هوفمان

مثال// لديك الصورة التالية بين بجدول كفاءة طرق الضغط:



شكل(76):صورة رقمية

الحل//

Table 1. Comparison of Compression Efficiencies for a 24-bit Image

Compression	Image bits/pixel	File Format	Size (kb)	(% of original)
none	24	TIFF	223,646	100
LZW	24	TIFF	243,829	109
run length	24	TGA	224,055	101
JPEG quality=75	24	JPEG	14,572	7
JPEG quality=50	24	JPEG	9,413	4

Table 2. Comparison of Compression Efficiencies for an 8-bit Image

Compression	Image bits/pixel	File Format	Size (kb)	(% of original)
none	8	TIFF	77,810	100
run length	8	PCX	79,028	102
LZW	8	GIF	59,582	77

طريقة ضغط الصورة vector quantization :

vector quantization هي عبارة عن معالجة لتحديد متجة لمموعة قيم صغيرة حيث تقسم الصورة إلى مجموعة من اللصور الجزئية أو البلوكات .

مثال//لديك جء الصورة التالي 4*4 يمكن تمثيلها D – 1 من المتجهات

$$\begin{bmatrix} 65 & 70 & 71 & 75 \\ 71 & 70 & 71 & 81 \\ 81 & 80 & 81 & 82 \\ 90 & 90 & 91 & 92 \end{bmatrix}$$

الحل//يمكن تمثيلها كمتجهات صفوف كالأتي

$$[\ row_1\ \ row_2\ \ row_3\ \ row_4\ ] = [\ 65\ \ 70\ \ 71\ \ 75\ \ 71\ \ 70\ \ 71\ \ 81\ \ 81\ \ 80\ \ 82\ \ 90\ \ 90\ \ 91\ \ 92\ ]$$

مثال// لديك صورة **256*256** , bit – **8** استخدم طريقة **vector quantization طريقة ضغط الصورة** علما أن عدد البلوك 4*4 والمتجه يتكون من 256 مدخل

$$\left(\frac{256\ \text{pixels}}{4\ \text{pixels / block}}\right) \quad \left(\frac{256\ \text{pixels}}{4\ \text{pixels / block}}\right) = 4.096$$

Blocks

أن بايت لكل 4*4 بلوك يعطي 4.096 بايت لعنوان المتجه و استخدام الحجم.**256*16**

4.096 + (256) (16) = 8.122 bytes for the coded file

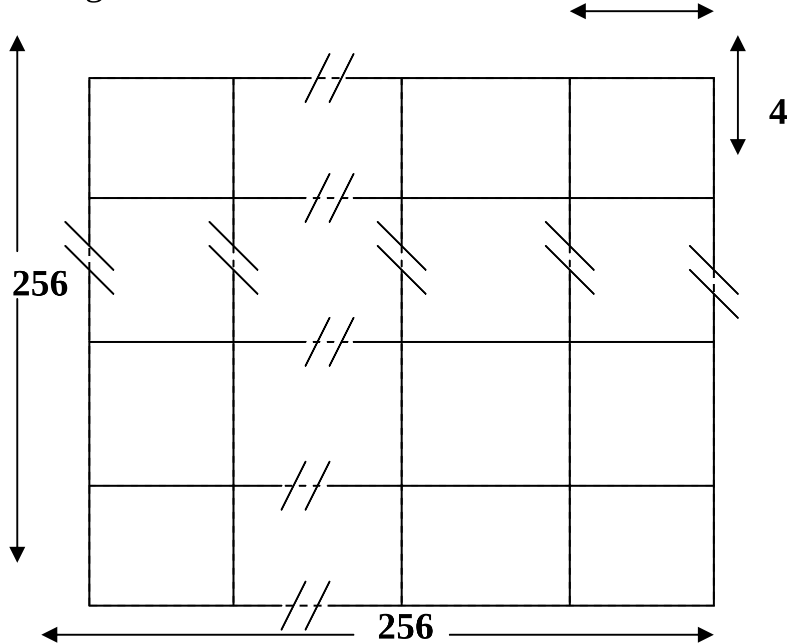
The original **8** – bit , **256*256** image contained

(256) (256) = 65.536 bytes

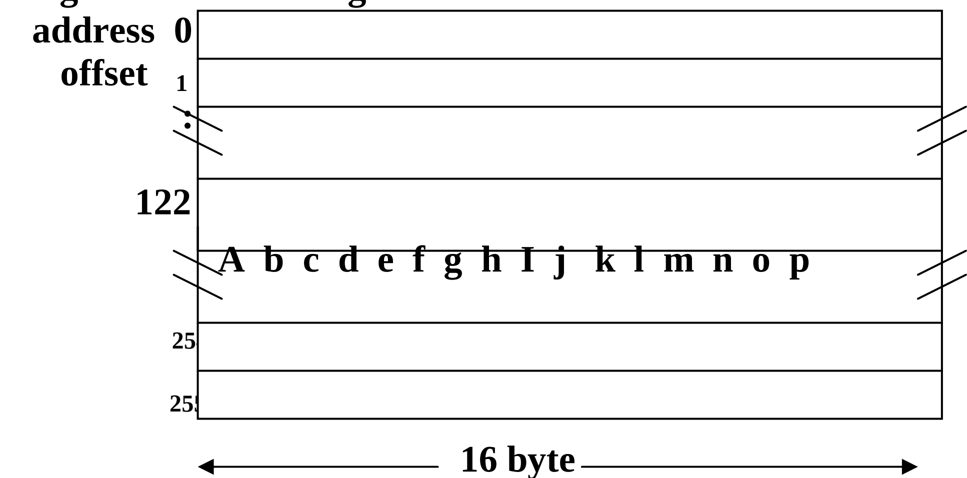
نحصل على نسبة الضغط

65.536 / 8.122 = 8 → 8:1 compression

the Quantizing with c codebook .



a. original 256*256 image divided into 4*4 blocks



b. codebook with 256 16 – byte entries

a	**b**	**c**	**d**
e	**f**	**g**	**h**
i	**j**	**k**	**l**
m	**n**	**o**	**p**

c. A sub image decompressed with vector # 122

شكل(77):طريقة **vector quantization** **وتنظيم** Quantizing with c codebook

الفصل السادس

بعض تطبيقات معالجة الصورة الرقمية

6 -1 مقدمة:

يعد مجال تحليل الصور مجالا متوسطا بين الرؤية بالحاسب ومعالجة الصور . يصعب إلى حد ما إيجاد حدود فاصلة بين هذه المواضيع الثلاثة (معالجة الصور ، وتحليلها والرؤية بالحاسب) .

إلا انه يمكن تقسيم العمليات التي يستخدم فيها الحاسب في هذا المجال إلي ثلاث مستويات :

1- عمليات ذات مستوي منخفض والتي تتضمن إزالة التشوه وتحسين التباين وزيادة حدة الصورة ، ويمكن وصف هذه العمليات بأنها تلك العمليات التي يكون دخلها صورة وخرجها صورة .

2- عمليات ذات مستوي متوسط والتي تتضمن تقسيم الصورة إلى مناطق أو عناصر ثم وصف هذه العناصر لاختزالها إلي تمثيل صالح للمعالجة بالحاسب ، كما تشمل أيضا عمليات التعرف علي عناصر محددة بالصورة ، ويمكن وصف عمليات المستوي المتوسط بكونها عمليات يكون دخلها صورة وخرجها خصائص وسمات مستخلصة من هذه الصورة ، مثال ذلك أطر العناصر وهوية تلك العناصر .

3- عمليات ذات مستوي عال وهذه تتضمن عملية فهم أو إدراك "making sense" لمجموعة من العناصر التي تم التعرف عليها وفي قمة عمليات هذا المستوي تأتي عمليات التعلم واكتساب المعرفة المرتبطة بالرؤية بالحاسب .

أن التداخل بين كل من معالجة وتحليل الصور يتمثل في عمليات التعرف علي مناطق أو عناصر معينة تنتمي للصورة . بالتالي يمكن وصف المعالجة الرقمية للصور علي أنها العملية التي يكون دخلها وخرجها صورة بالإضافة إلي عمليات استخلاص خصائص وسمات الصورة وحتى التعرف علي العناصر التي تنتمي للصورة.

بالموازاة مع التطبيقات الخاصة بالفضاء ظهرت تطبيقات طبية تتمثل في التصوير الطبي وخاصة بعد ابتكار "المسح بالحاسب computerized tomography, CT" والتي يمكن عن طريقها رسم صور مجسمة ثلاثية الأبعاد للمريض أو لهدف ما عن طريق مصدر للأشعة السينية وحلقة من مجسات الأشعة السينية تحيط بالهدف حيث يتم تحريك مصدر الأشعة دائريا ثم تحريك المجسات خطوة بخطوة رأسيا .

من التطبيقات الشيقة الأخرى والتي طورت منذ الستينات وحتى الآن ، تحسين التباين أو تحويل مستويات الشدة intensity إلي ألوان لتسهيل فهم واستيضاح صور الأشعة السينية والصور الأخرى في الصناعة وفي المجالات الحيوية .

2-6 التعرف علي بعض المجالات التطبيقية في موضوع المعالجة الرقمية للصور:

يمكن تقسيم الطيف الكهرومغناطيسي حسب طاقة الفوتون إلى النطاقات الموضحه في الشكل الاتي .



شكل(78): تقسيم الطيف الكهرومغناطيسي

1-2-6 التصوير بأشعة جاما :



شكل(79):صورة اخذت بتقنية positron emission tomography

أخذت الصورة في الشكل اعلاه بتقنية ال positron emission tomography والتي يتم فيها حقن المريض بنظير مشع يشع جسيمات البوزيترون و عند تقابل البوزيترون و الإلكترون

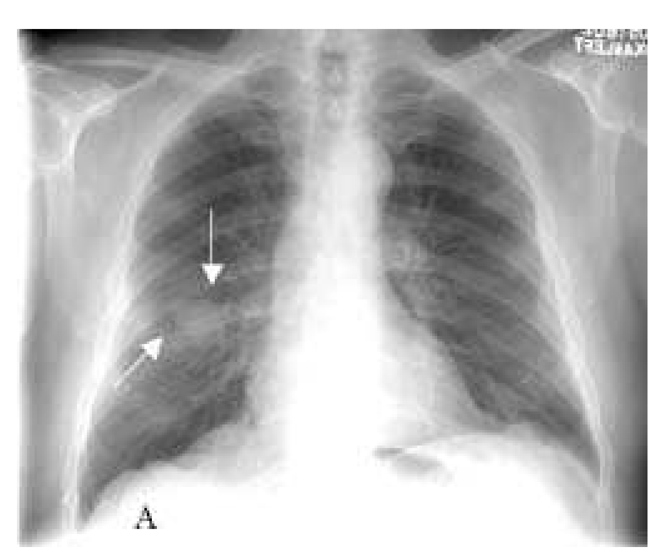
يتلاشيان و ينتج شعاعان من نوع جاما ثم بعد ذلك يتم استشعار هذه الأشعة من خارج الجسم عن طريق حساسات خاصة تدور حول الجسم لتكوين صورة ثلاثية الأبعاد . كما يمكن التصوير عن طريق استقبال أشعة جاما من المصدر/الجسم المراد تصويره .

6-2-2 التصوير بالأشعة السينية:

من أشهر طرق التصوير بالأشعة السينية استغلال قدرة النفاذ للأشعة السينية خلال الأجسام و استقبال الطاقة النافذة علي فيلم حساس ، حيث تختلف درجة تأثر الفيلم بطبيعة الجسم المراد تصويره .

من الممكن تكوين صور ثلاثية الأبعاد بالأشعة السينية عن طريق تقنية ال Computerized Axial Tomography (CAT) حيث يتم احاطة الجسم المراد تصويره بحلقة من الحساسات ثم تحريك مصدر للأشعة السينية علي هذه الحلقة ليؤثر علي العنصر المقابل في الحلقة، وبذا يتم تصوير مقطع في الجسم ثم بتحريك الحلقة ككل في اتجاه عمودي يتم أخذ مقاطع أخري. بعد ذلك و عن طريق الحاسب يتم تكوين صورة ثلاثية الأبعاد للجسم .

أن للأشعة السينية تطبيقاتها في مجال الصناعة حيث يمكن اكتشاف الكسور أو الشروخ في دائرة الكترونية . ومن الممكن تصوير مصدر الأشعة السينية نفسها .



http://www.dcmsonline.org/jax-medicine/2003journals/lungcancer/

شكل (80) : صوره للقفص الصدري



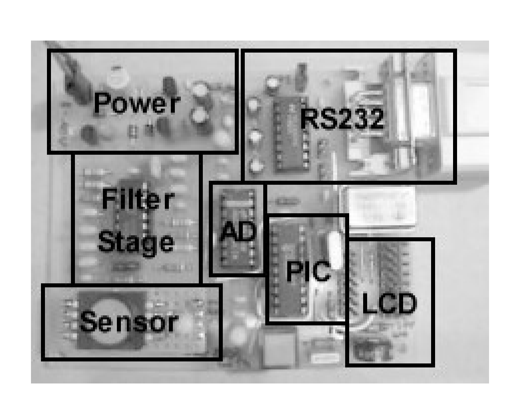
http://cpmcnet.columbia.edu/dept/radiology/eastside/cAt.html
شكل (81): مقطع في صورة ثلاثية الأبعاد مصورة بتقنية ال CAT

6-2-3 التصوير في النطاق فوق البنفسجي:

يعتمد التصوير في هذا النطاق علي ظاهرة فيزيائية وهي أنه عند سقوط الأشعة فوق البنفسجية علي مادة فلورسية فأنها تشع ضوء وذلك لأن الأشعة فوق البنفسجية نفسها غير مرئية. كما يمكن أيضا أخذ صور لمصدر للأشعة فوق البنفسجية.

6-2-4 التصوير في النطاق المرئي:

وهي أكثر النطاقات المستخدمة في التصوير، ويوضح الشكل الآتي صور لبعض الدوائر الالكترونية.





شكل(82): صور لبعض الدوائر الالكترونية.

6-2-5 التصوير في نطاق الموجات الميكروية:

يعتمد التصوير في هذا النطاق عادة علي الرادار ، ويشبه التصوير هنا طريقة التصوير بالكاميرا العادية ذات الفلاش ، حيث يتم إضاءة المساحة المطلوب تصويرها بنبضات الموجات الميكروية، ثم يتم أخذ لقطة للمشهد عن طريق الموجات المنعكسة والتي يتم استقبالها بواسطة هوائي خاص، بعد ذلك وبواسطة المعالجة الرقمية بالحاسب يتم تسجيل هذه الصور .

6-2-6 التصوير في نطاق الموجات الراديوية:

يعد التصوير الطبي والفلكي أهم مجالات التصوير في هذا النطاق. تستخدم موجات الراديو في التصوير الطبي فيما يسمي بالتصوير بالرنين المغناطيسي. حيث يتم وضع المريض علي مغناطيس قوي ثم توجه نبضات قوية من موجات الراديو إلي الجزء المطلوب تصويره، بعد ذلك يتم التقاط الموجات المنعكسة من الأنسجة ويتم تحديد مكان وقوة النبضات المنعكسة باستخدام حاسب رقمي.



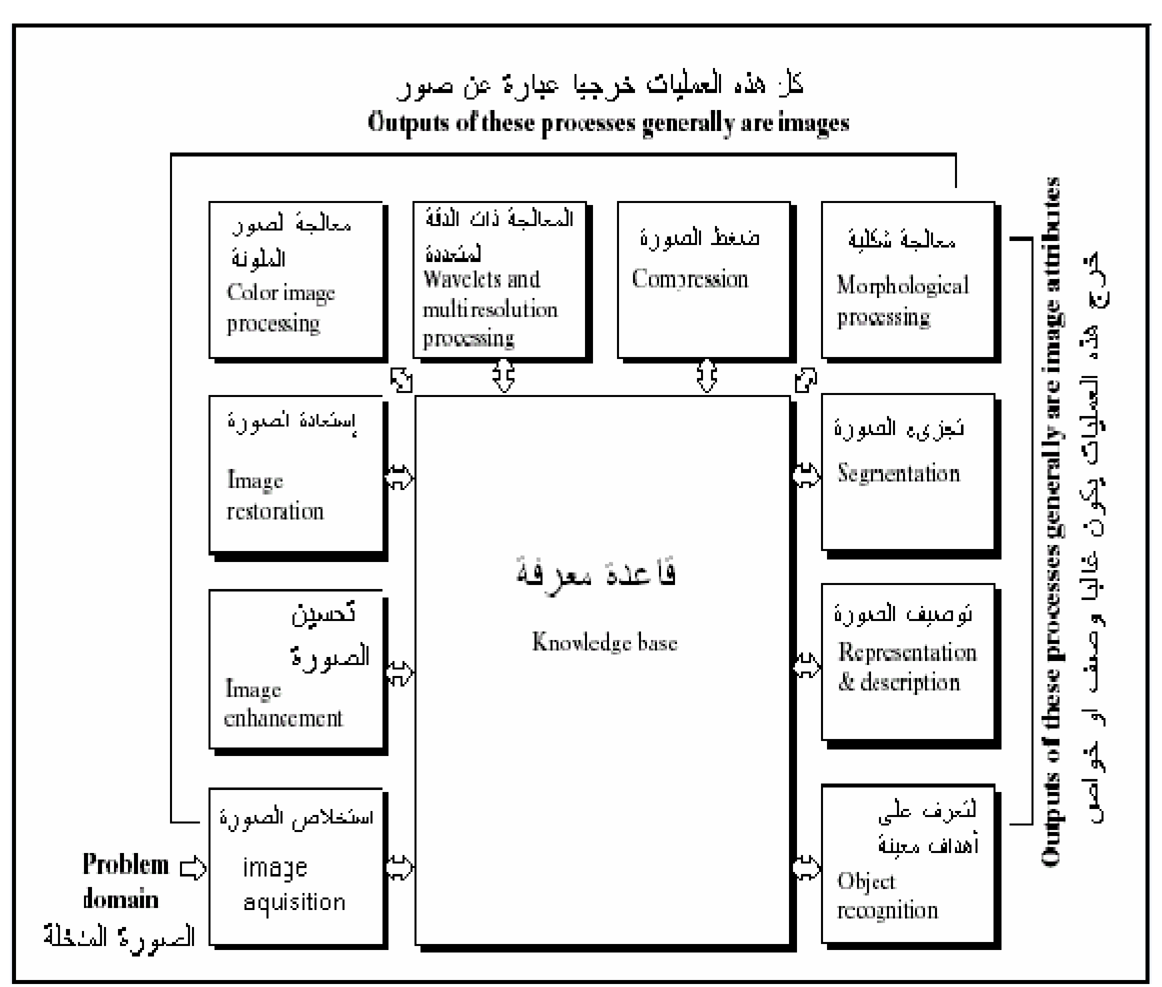
http://www.lowbackpain.com/faq.shtml



http://brighamrad.harvard.edu/Cases/bwh/images/118/MR2Stud

6-3 تطبيقات المعالجة الرقمية للصور:

يمكن تحديد تطبيقات معالجة الصورة من خلال العمليات التالية على الصور الرقمية الموضحة بالشكل ادناة:



شكل(84):العمليات التي تجرى على الصور الرقمية

1- الحصول علي الصورة أو استخلاصها image acquisition
دراسة طرق الحصول علي الصورة عن طريق أجهزة خاصة . يعد أبسط أشكال هذه العملية هو مجرد الحصول علي صورة رقمية مباشرة من جهاز حاسب .

2- تحسين الصورة image enhancement
ويقصد بها تلك العمليات التي يتم بها إيضاح بعض التفاصيل المهمة في الصورة أو التركيز علي بعض الصفات والسمات موضع الأهمية من الصورة . من أبسط الأمثلة علي هذه العملية هو زيادة التباين فى الصورة وذلك فقط لأن البشر يرون في ذلك وضع أفضل وأيسر لرؤية وفهم محتويات الصورة .

3- استرجاع أو استعادة الصورة image restoration
تهتم هذه العملية أيضا بتحسين مظهر الصورة ولكن عملية التحسين هنا تعتمد علي بعض النماذج الرياضية أو الإحصائية لمعالجة الصورة . بينما تعتمد عملية تحسين الصورة enhancementعلي متطلبات بشرية مثل مقدار جودة الصورة بالنسبة لمشاهدها .

4- معالجة الصور الملونة

ترجع أهمية هذه العملية إلي كثرة استخدام الصور الملونة الآن علي شبكة الإنترنت والحاجة دوما إلي معالجتها بالإضافة إلي انه يمكن استخلاص بعض السمات والخصائص من الصورة بناء علي اللون .

5-المعالجة ذات الدقة المتعددة باستخدام المويجات wavelets

وهو حجر الأساس في تمثيل الصورة بدرجات متفاوتة من الإيضاح resolution ويستخدم عادة في ضغط الصور والبيانات .

6- الضغط compression

يستخدم في اختزال حجم الذاكرة المطلوب لتخزين الصورة وبصورة أخري يؤدي هذا أيضا إلي اختزال عرض النطاق band width المطلوب لإرسال الصورة .

7- معالجة التشكل morphological processing

يهتم باستخلاص مكونات الصورة والتي تكون مفيدة في تمثيل وتوصيف شكل معين بالصورة وتعد هذه العملية أول العمليات التي تصادفنا والتي يكون خرجها هو خصائص أو سمات للصورة .

8-التقسيم أو التجزىء segmentation

ويهتم بتقسيم الصورة إلي مكوناتها الجزئية أو إلي عناصر ويعتبر من أهم العمليات علي الصور لأن الخطأ في عملية التقسيم يعني فشل في كل التطبيقات التي تعتمد علي ذلك ومنها مثلا التعرف عل الصور image recognition .

9-التمثيل والوصف

يلي عملية التقسيم segmentation ، عملية التمثيل ويتم فيها تمثيل مكونات الصورة بأحد طريقتين : تمثيل إطاري boundary أو تمثيل مساحي regional ومن المعروف أن التمثيل الإطاري يهتم بالشكل الخارجي لعناصر الصورة بينما يهتم التمثيل المساحي بالخصائص الداخلية للصورة . أما عملية الوصف description فتهتم باستخلاص خصائص وصفات الصورة والتي تكون ذات أهمية في عملية التصنيف classification .

10-التعرف علي الصور recognition

تهتم هذه العملية بإيجاد مسمي لعنصر ما في الصورة بناء علي وصفه وسماته .

4-6 التداخل الترددي وأشكال موير:

من المعروف أنه يمكن تمثيل الدالة المحدودة زمنيا بمجموعة من المركبات الجيبية حيث يكون أعلي تردد لهذه المركبات الجيبية محدودا ، وحسب نظرية شانون فإنه إذا تم أخذ عينات من هذه الدالة بمعدل أكبر من أو يساوي ضعف نطاقها الترددي فإنه يمكن استرجاع هذه الدالة ، ولكن أذا تم أخذ العينات بمعدل اقل من المذكور آنفا فإنه يحدث ما يسمي بالتزييف aliasing حيث تتداخل تكرارات النطاق الترددي الممثل للدالة وبالتالي لا يمكن استرجاعها ، وبالنسبة للصورة فإن معدل أخذ العينات هو معدل أخذ العينات في كل من الاحداثيين x,y .

من المعروف أنه من المستحيل تحقيق نظرية شانون عمليا لأننا نعمل دائما علي دوال محدودة في الزمن. ولكن من الممكن أن نقوم بتحويل الدالة الغير محدودة في الزمن إلي دالة محدودة في الزمن عن طريق دالة البوابة (gate function) ولكنها لسوء الحظ هي نفسها غير محدودة في النطاق الترددي. و بالتالي محاولتنا لتحديد زمن دالة ذات نطاق ترددي محدود يجعلها غير محدودة في التردد. و لذا يكون الحل الأساسي هو تقليل قيمة المركبات ذات الترددات العالية ليمكن إهمالها فتكون الدالة محدودة في النطاق الترددي . يمكن رؤية تأثير التداخل الترددي aliased frequency تحت ظروف معينة في شكل يسمي بنموزج موير moiré pattern .

يمكن فهم التكبير علي أنه مضاعفة لعدد العينات كما يمكن فهم التصغير علي أنه إنقاص لعدد العينات ، إلا أنه لكى نفرق بين عملية أخذ العينات (Sampling) وكل من التكبير والتصغير فإنه يمكننا القول بأن التكبير والتصغير يتم علي الصور الرقمية فقط . يتطلب التكبير عمليتين :

1) إيجاد أماكن لمجموعة من عناصر الصورة الجديدة .
2) إيجاد قيمة لمستوي الرمادي لكل من هذه العناصر الجديدة .

يتم هذا بمجموعة من الطرق أبسطها هو تكرار ونسخ العنصر كما تم شرحه سابقا حيث يتم مضاعفة حجم الصورة بعدد صحيح من المرات ثم يتم نسخ المستوي الرمادي للعنصر المجاور لذلك المنشأ حديثا . ولكن عيب هذه الطريقة أن المضاعفة تتم بعدد صحيح فقط . من الطرق الأخرى طريقة تسمي nearest neighbor interpolation .

فمثلا إذا أردنا تكبير صورة ذات حجم 500×500 إلي أخرى ذات حجم 750×750 فإننا نكون شبكة خالية بالحجم الجديد 750×750 ثم نطابقها علي الصورة الأصلية ، بالطبع تكون المسافات بين عناصر الصورة الجديدة صغيرة عن تلك فى الصورة الأصلية ، لذا فإننا نعطي لكل عنصر في الصورة الجديدة قيمة مستوي الرمادي للعنصر الأقرب لها مسافة في الصورة الأصلية بعد مطابقة الشبكتين .

توجد طريقة تسمي bilinear interpolation والتي تستخدم أقرب أربع نقاط . لذا فإذا افترضنا أن مستوي الرمادي للنقطة المطلوبة هو $v(x\ ,y)$ فإنه يمكن حسابه من المعادلة :

$$v(x\ ,y) = ax + by + cxy + d\ldots\ldots\ldots(58)$$

حيث الثوابت الأربعة غير معلومة ويمكن حسابها من الأربع معادلات المماثلة لأقرب أربع نقاط في الصورة الأصلية . أما بالنسبة للتصغير فإنه من أبسط الطرق أن يتم حذف نصف عدد الأعمدة والصفوف (إما الزوجية أو الفردية) . ومن الممكن أيضا استخدام طريقة nearest neighbor interpolation أو bilinear interpolation .

ولتقليل تأثير ال aliasing يمكن عمل ما يسمى بالتبهيت (جعل الصورة باهتة) blurring، ويمكن أيضا عمل عملية interpolation بعدد أكبر من النقاط المجاورة والتي تعطي نتائج أفضل إلا أنه بالطبع يحتاج مجهود حسابي أكبر .

5-6 بعض العلاقات الأساسية بين مجموعات العناصر:

1- جيران العناصر

إذا كان لدينا عنصرا p(x,y) فإن الأربع نقاط المجاورة رأسيا وأفقيا تسمي 4-neighbor و هم :

$$(x+1,y), (x-1,y), (x,y+1), (x,y-1) \ldots\ldots\ldots.(59)$$

تسمي هذه المجموعة بال N4(p) ، وبالطبع النقاط التي علي نطاق الصورة تقع بعض جيرانها خارج الصورة . تسمي النقاط المجاورة ل p(x,y) علي الاتجاهات الفرعية (القطرية) بالجيران القطرية Diagonal neighbors, Nd(p) وهم :

$$(x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1) \ldots\ldots\ldots(60)$$

تسمي مجموعة النقط التي تحوي كل من N4(p) ، Nd(p) بالجيران الثمانية 8-neighbors (N8(p)).

2- التجاور والاتصال والمناطق والأطر

يعد الاتصال بين عناصر الصورة مفهوم هام لتبسيط تعريف المناطق والأطر . والآن يمكننا القول بان عنصرين في الصورة يكونان متصلين إذا كانا :

1) متجاورين

2) المستويان الرماديان لكل منهما يحققان شرطا معينا و ليكن التساوي .

أما بالنسبة للتجاور فإنه يمكننا تعريف ثلاث أنواع للتجاور:

1) التجاور الرباعى 4-adjacency

p,q 4-adjacent إذا كان q Є N4(p) ، يقال أن التجاور من النوع الرباعى المستوي Vلتكن لكل منهما ينتمي لمجموعة محددة من مستويات الرمادي .

2 8-adjacency التجاور الثمانى

، يقال أن التجاور من النوع الثمانى المستوي $q \in N8(p)$ إذا كان p,q 8-adjacent
لكل منهما ينتمي لمجموعة محددة من مستويات الرمادي

3 - m-adjacency التجاور العام
p,q m-adjacent يقال أن التجاور من النوع العام إذا كان :
المستوي الرمادي لكل منهما ينتمي إلى
-$q \in N4(p)$ أو ،
- $q \in Nd(p)$ ، وعناصر الصورة $N4(p) \cap N4(q)$ ليس لها مستوي رمادي ينتمي

- يقال أن مجموعتين من عناصر صورة ما s1,s2 متجاورين إذا وجد بعض عناصر الصورة في s1 مجاورة لبعض عناصر الصورة في s2 . ونعني في السياق التالي بالتجاور 8,4,m-adjacency .

- يعرف المسار الرقمي أو المنحني من عنصر (x,y) إلي عنصر (s,t) بتتابع العناصر التالي :

(x0,y0), (x1,y1) ......(xn,yn)………………(61)

حيث :
متجاورين (xi-1,yi-1) , (xi,yi) ، وكل عنصرين (x0,y0)=(x,y), (xn,yn)=(s,t)
فإن (x0,y0)=(xn,yn) في هذه الحالة بطول المسار . إذا كان n ويسمي 1<=i<=nلكل
المسار يكون مغلقا .

- يسمي المسار ب 8, m, 4 path إذا كان التجاور 8,m,4 adjacency ويلاحظ غياب اللبث في المسار m-path .

- لنفترض أن s هي مجموعة من العناصر في صورة و أن p,q نقطتين تنتميان إلي s ، يقال أن p,q متصلتين connected إذا كان هناك مسار path بينهما يتكون كاملا داخل s .

- لأي نقطة p تسمي مجموعة النقاط التي تتصل بها بمجموعة الاتصال connected component.

- إذا كانت s تكون فقط هذه المجموعة فإن s تسمي مجموعة متصلة connected set.
- إذا كانت R مجموعة من العناصر في صورة فإننا نسمي R منطقة region إذا كانت R مجموعة متصلة.

- نطـاق منطقـة R (boundary, border, contour) هـو مجموعـة العناصـر فـي المنطقة R والتي يكون كل أو احد جيرانها لا ينتمي إلي R .

- إذا كانـت R هـي الصـورة ككـل فـإن نطاقهـا هـو الصـف الاول والأخيـر والعمـود الأول والأخير.

- يمكـن فهـم الحافـة edge علـي أنهـا تغيـر مفـاجيء أو انقطـاع فـى الكثافـة intensity discontinuities بينما الأطرى على أنها مسارات مغلقة .

6-6 مقاييس المسافة:

لأي ثلاث نقاط p,q,z لها الإحداثيات (x,y), (s,t), (v,w) تكون D دالة مسافة إذا كان :

- $D(p , q)>=0, D(p , q)=0$ if $p=q$

- $D(p , q)=D(q , p)$
- $D (p , z)<=D(p , q)+D(q , z)$

يمكن تعريف علي أنها Euclidean distance:مسافة اقليدس

$$De (p , q)=[(x-s)^2+(y-t)^2]^{1/2} \ldots\ldots (62)$$

لهذا النوع من دوال قياس المسافة فإن عناصر الصورة التي تـقـع علي بـعد ثابت نقـطـة (x,y) ، تـمثل قرصـا مـركزه (x,y) ونـصف قطره r ، يمكـن تعريـف الـcity-block distance علي أنها :

$$D4 (p , q) = |x-s| + |y-t| \ldots\ldots (63)$$

في هذه الحالة فإن عناصر الصورة التي تقع علي بعد ثابت r من (x,y) مقاسا بهذه الدالة تمثل معينا مركزه (x,y) ، ويمكن تعريف ال chessboard distance على أنه:

$$D8 (p , q)=\max(|x-s|, |y-t|) \ldots\ldots (64)$$

في هذه الحالة فإن عناصر الصورة التي تقع علي بعد ثابت r من (x,y) مقاسا بهذه الدالة تمثل مربعا مركزه (x,y) وطول ضلعه 2r ، نلاحظ أن D4,D8 بين هذه النقط تعتمد فقط علي أحداثيات النقط ، بينما إذا أخذنا في الحسبان m-adjacency فإن قيمة هذه النقط والنقط المجاورة لها تؤثر علي هذه المسافة . فمثلا حسب قيم جيران النقطتين(p,q) فإنه يمكننا إيجاد أكثر من m-path وكل بطول مختلف . Dm علي أنها أقصر m-path بين النقطتين .

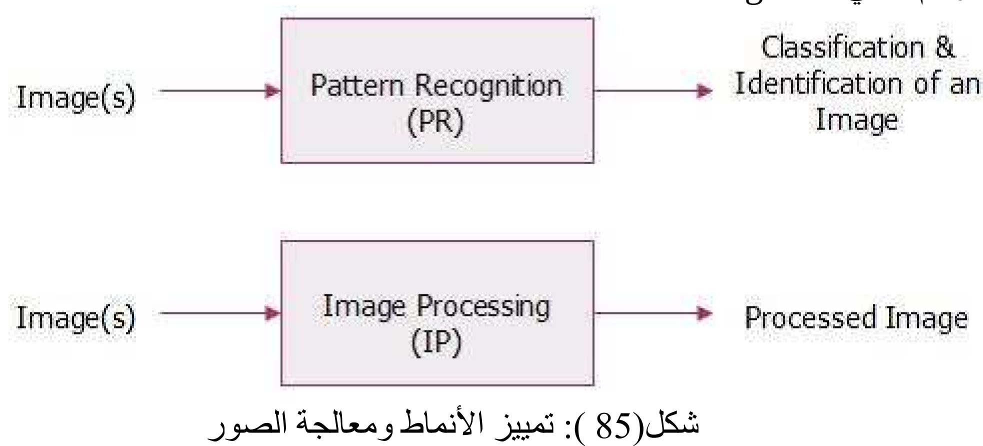
إذا كان H معامل خطي دخله وخرجه صورة ، يقال أن H إذا تحقق لأي صورتين f , g ولأي ثابتين a,b

$$H\ (af,bg) = aH(f) + bH(g)..............(65)$$

بمعنى آخر فإن نتيجة استخدام معامل خطي علي مجموع صورتين (بعد ضربهما في ثابتين عددين) هي نفسها نتيجة استخدام المعامل مع الصورتين فرادي (بعد ضربهما في نفس الثابتين العددين) ، يسمي المعامل الذي لا يحقق الشرط السابق بمعامل غير خطي (لا خطي) .

7-6 تمييز الأنماط ومعالجة الصور
Introduction to Pattern Recognition and Image Processing

وهو فرع من فروع الذكاء الاصطناعي، ومعنى تمييز الأنماط ومعالجة الصور يوضحهما الرسم التالي:Block diagrams



شكل(85): تمييز الأنماط ومعالجة الصور

بمعنى أن أي برنامج لتمييز الأنماط يدخل له صورة فيعطي تصنيف أو تعرف للصورة، وأي برنامج للعمليات على الصور يدخل له صورة فيعطي صورة تمت بعض العمليات عليها.
أن أي طريقة للتعرف على الأنماط أو أي شئ في العالم لابد من أن يسبقها مرحلة تعلم لهذه الأنماط وهذه الأشياء، إذن مراحل التعرف على أي نمط مرحلتين:

1- مرحلة التعلم learning
2-مرحلة التصنيف classification or recognition

تمييز الأنماط أو التعرف على النماذج Pattern recognition هو أحد فروع علم تعلم الألة وبشكل عام الذكاء الإصطناعي ، تهدف البحوث والتقنيات الخاصة بهذا العلم إلى إيجاد أو تطوير تقنيات للتعرف على أنماط أو هياكل محددة في الإشارات الرقمية، حيث يمكن للإشارة أن تمثل صورة تحوي حرف مكتوب أو مقطع موسيقي أو مقطع كلامي يمثل كلمة أو حتى نص حاسوبي، ويمكن أن يكون النمط المطلوب التعرف عليه هو الحرف الذي تحويه الصورة أو الآلة المستخدمة في المقطع الموسيقي أو الكلمة الملفوظة في المقطع الكلامي .

الهيكل العام لنظم التعرف على النماذج يتكون من:

1-اكتساب المعلومات: يتم فيها الحصول على الدخل الذي نريد التعرف عليه من المعلومات المستخدم.

2 - معالجة الإشارة قبل بدء التعرف: في هذه المرحلة نقوم بإزالة التشويش من الإشارة وتحويلها إلى شكل نظامي Normal Form باستخدام التقييس Scaling وعمليات أخرى بسيطة. الهدف هو حصول على إشارة "نظيفة" تسهل على باقي المراحل العمل.

3-استخلاص الخصائص المميزة: في هذه المرحلة يتم إيجاد صفات وخصائص من الإشارة تساعد على تحديد النموذج (النمط) الذي تمثله.

فمثلاً في مجال التعرف على الكلام، فإن المعلومات اللغوية في الإشارة هي التي تحدد الكلمة، وليست المعلومات التي تحدد المتكلم أو حالته النفسية. لو استطعنا استخلاص المعلومات اللغوية بشكل دقيق، يصبح التعرف أسهل (إذ نكون قد حذفنا معلومات أخرى غير مفيدة في التعرف).

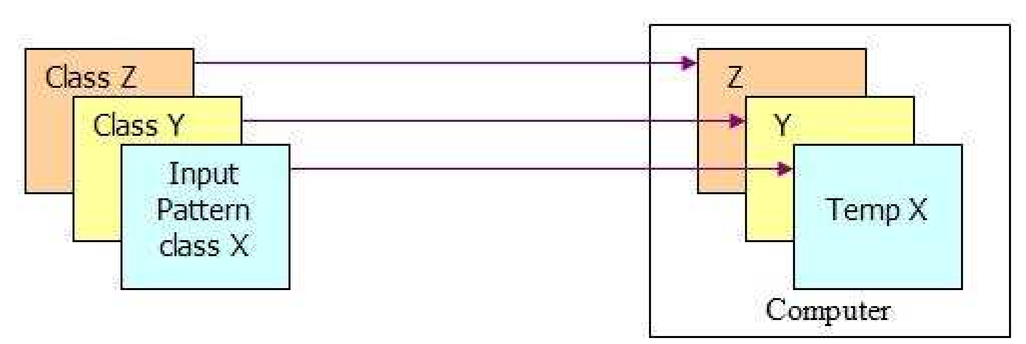
4-التصنيف: هنا الدخل هو شعاع من الخصائص المميزة، وعلينا تحديد أي من النماذج المخزنة يمثلها هذا الشعاع. هناك عدة تقنيات كالشبكات العصبية وغيرها.

يوجد أربعة طرق أساسية مستخدمة في عالم تمييز الأنماط ألا وهي

1. Template-Matching and Correlation Method.
2. Statical Approach.
3. Syntactic and Structural Approach.
4. Neural Networks Approach.

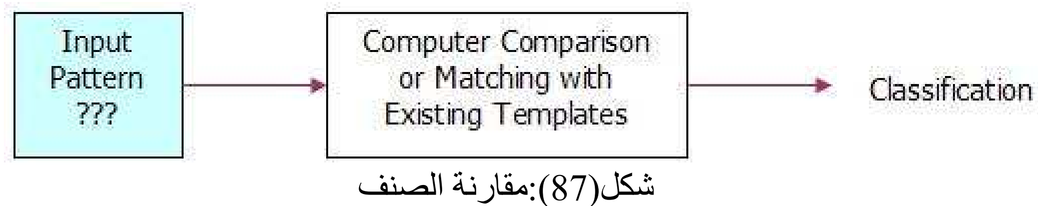
الطريقة الأولى (Template-Matching and Correlation Method):

مرحلة التعليم في هذه الطريقة تقوم على تخزين مجموعة من القوالب أو النماذج Prototypes، قالب من كل صنف في الحاسوب كما يوضح الرسم:



شكل(86): على تخزين مجموعة من القوالبTemplates

وفي مرحلة التصنيف تقارن الصورة الداخلة Input pattern مع Templates الخاص بكل صنف فإن كانت نتيجة مقارنتها مع الصنف س أكبر من نتيجة مقارنتها مع الصنف ص فإنها تصنف ضمن الصنف س و هكذا.



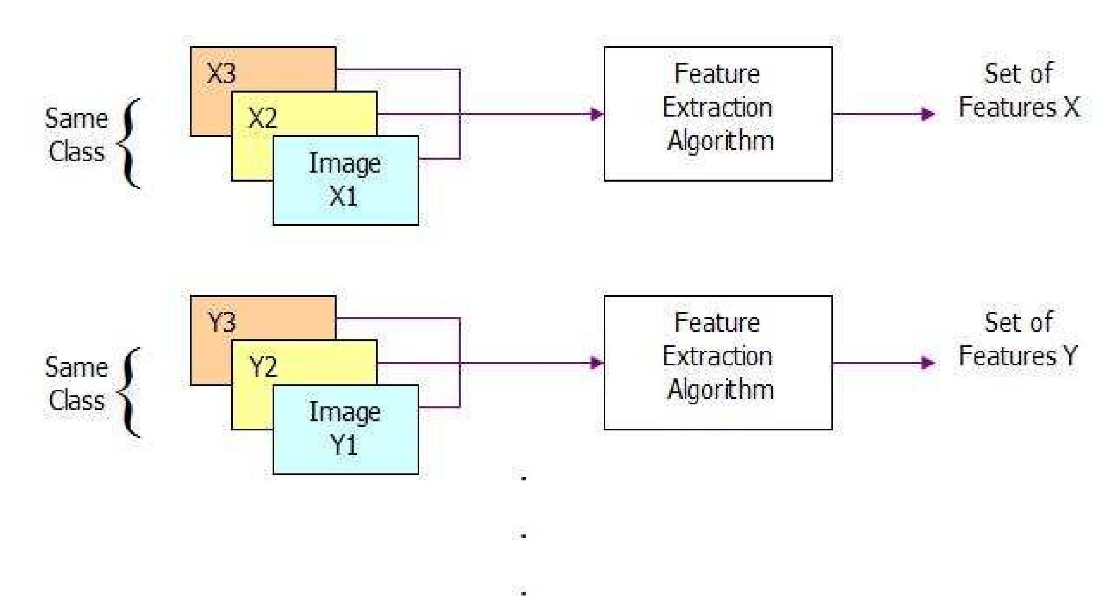
شكل(87):مقارنة الصنف

تخزن الصورة الداخلة على شكل مصفوفة وتقارن مع القوالب الموجودة في الجهاز pixel by pixel وتعطي قيمة للمقارنة.

تعتبر هذه الطريقة طريقة سهلة وبدائية جداً، الصعوبة الوحيدة في هذه الطريقة هي الاختيار الجيد للقوالب من كل صنف بالإضافة إلى تحديد معايير المقارنة وخصوصاً لو كانت الصورة الداخلة تحمل تشوهات! ،فمثلاً لو استخدمنا هذه الطريقة للتعرف على المجرميين، لابد أن نأخذ لكل مجرم عدة لقطات كي تخزن على جهاز الحاسوب: لقطتان جانبيتان واحدة من كل جهة، لقطة أمامية، ولقطتان بزاوية نظر 45 درجة عن الكاميرا. ولكم أن تتخيلوا المساحات التخزينية اللازمة لكل هذه القوالب!

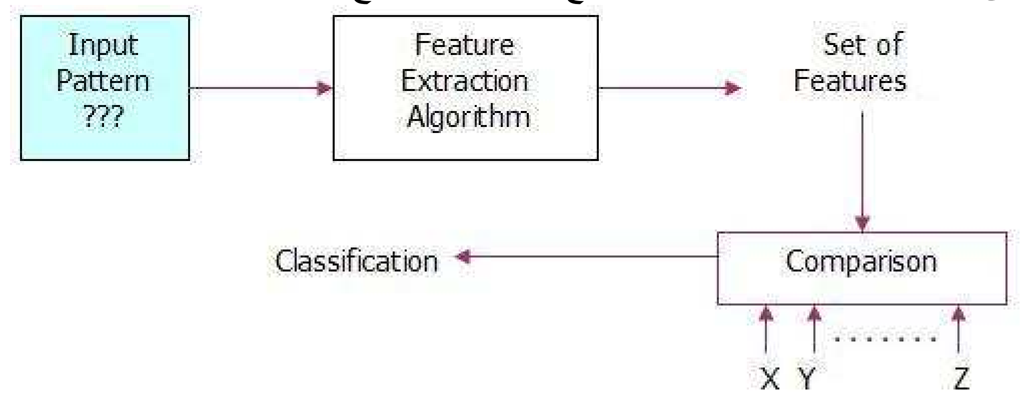
الطريقة الثانية (Statical Approach):

في هذه الطريقة، يوصّف كل pattern بواسطة مجموعة من الخصائص set of featuresوالتي من الممكن أن نعبر عنها بقيم حقيقية. في مرحلة التعلم: يقدّم كل نمط patternكمتجه من الخصائص feature vector كما توضح الصورة:



شكل(88):تمثيل نمط pattern كمتجه من الخصائصfeature vector

أما في مرحلة التعرف أو التمييز أو التصنيف، فهذه عادة تتم عن طريق تقسيم مساحة الصورة إلى مناطق مجزأة، كل منطقة تقارن مع صنف كما توضح الصورة:

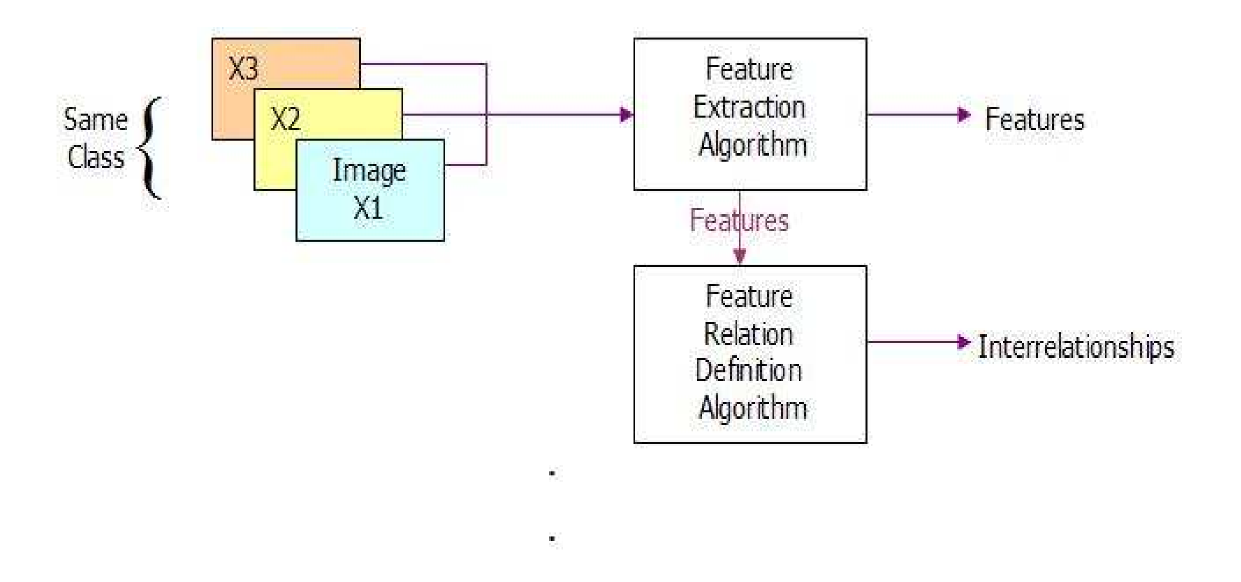


شكل(89): تقسيم مساحة الصورة إلى مناطق مجزأة

تعتمد على خصائص الصورة التي نخزنها في مرحلة التعلم :اللون، الشكل، الدوران، المنطقة السفلى، المنطقة العليا.... ألخ. وكذلك يتم التعرف على الصورة، تقسم الصورة إلى أجزاء وكل جزأ نقارن الخصائص الموجودة فيه مع خصائص الصنف المخزنة وهكذا.
الصعوبة هنا هي في اختيار مجموعة الخصائص لكل فئة وقواعد القرار في التعرف على النمط

الطريقة الثالثة (Syntactic and Structural Approach)

في هذه الطريقة لا نكتفي فقط بالقيم الرقمية لخصائص كل صنف، ولكن نضيف عليها العلاقات البينية Interrelationships or Interconnection of Features بين هذه الخصائص في كل صنف والتي تتيح لنا معلومات هيكلية ضرورية في التعرف على الأنماط.
آخر الدراسات في هذا المجال توصلت إلى أن أقوى طريقة للتعرف على الأنماط هي الطريقة التي تجمع بين Statistic pattern recognition approach مع Syntactic pattern recognitionكطريقة واحدة تسمى.Syntactic-Semantic approach
في مرحلة التعلم في هذه الطريقة يمثل النمط عادة كشجرة tree أو رسم بياني graph أو سلسلة حرفية string من العناصر الأولية primitives والعلاقات بينها relations



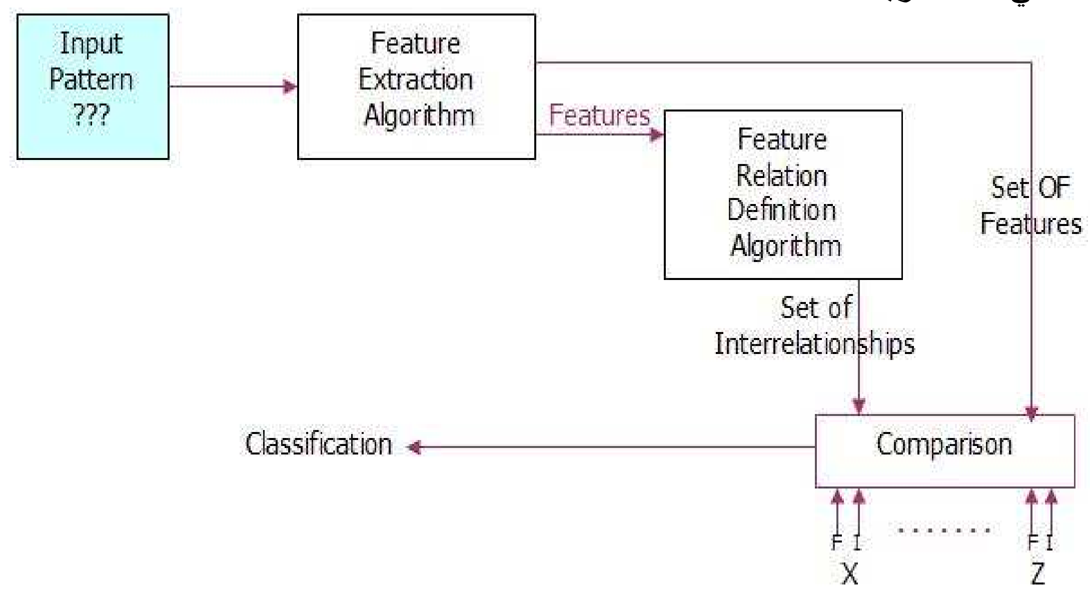
.

.

.

شكل(90): الطريقة الثالثة Syntactic and Structural Approach

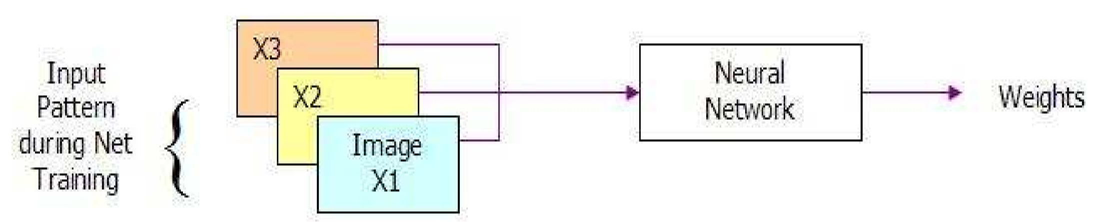
وعملية اتخاذ القرار في مرحلة التعرف أو التصنيف هي عبارة عن عملية تحليل أو بمعنى آخر برنامج تعريب parsing procedure، وأعلى نسبة مقارنة ناتجة من مقارنة الصورة المدخلة مع كل شجرة (tree أو graph أو string على حسب التمثيل المعتمد في التطبيق) مخزنة تحدد الصنف الذي تنتمي إليه الصورة المدخلة! الشكل التالي يوضح عملية التصنيف في هذه الطريقة:



شكل(91): عملية التصنيف الطريقة الثالثة
Syntactic and Structural Approach

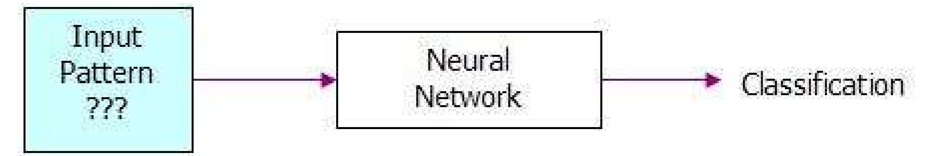
الطريقة الرابعة (Neural Networks Approach)

الشبكات العصبية علم قائم بحد ذاته اهتم بـه العلماء لسنوات عديدة بهدف الوصول إلى طريقة أشبه ما تكون بطريقة الإنسان في التعرف على الأنماط. ولا يسعنا شرح مفاهيم العلم في هذه الوهلة ولا بسلسلة دروس متكاملة، ولكن باختصار يقوم على استخدام المعالجة المتوازية للبيانات في وقت واحد، هذه المعالجة تتم في معالجات أو وحدات units أو طرفيات- nodesوكلاهم بمعنى واحد- تتصل ببعضها البعض عن طريق وصلات ذات أوزان weightsوالتي ضبطت أثناء عملية تدريب الشبكة.
وفي مجال تمييز الأنماط مجموعة من الصور patterns تدخل إلى الشبكة العصبية فتقوم الشبكة العصبية بضبط أوزانها طبقاً لميكانيزم معين وعمليات طويلة:



شكل(92): الطريقة الرابعةNeural Networks Approach

بعد ذلك وفي مرحلة التصنيف يقدم للشبكة pattern وبناء على الأوزان فيها تقوم بتصنيف هذا النمط



شكل (93):التصنيف بالطريقة الرابعةNeural Networks Approach

8-6 تشفير صورة باستخدام خوارزمية جديدة لتحديد وتشفير حواف ألوان الصورة:

ان مرحلة الحصول على بيانات الصورة وتحليلها أو تحويلها إلى الصيغة أو الشكل الذي من خلاله نستطيع تطبيق تتم من الأتي:

أ- تحليل الصورة وإجراء عملية مسح (SCANNING) لها للحصول على بيانات كل عنصر عرض في الصورة (Pixel) وعادة ما تكون موضوعه بالنظام العشري (Decimal) .

ب-نقوم بعملية تحويل صيغة بيانات الصورة من النظام العشري (Decemal System) إلى النظام الثنائي (System Binary) .
حيث إن كل عنصر (Pixcel) في الصورة يتم تحويله إلى 24 بت في النظام الثنائي ، وبما إن كل عنصر عرض يتكون من ثلاث ألوان أساسية مشتركة في تكوينه وهي الأحمر،الأخضر،الأزرق (نظام ألوان RGB) .

فان لكل لون من هذه الألوان يحجز 8 بتأت خاصة به وبالنتيجة فان الشدة اللونية لكل لون سوف تتراوح ما بين (0_ 255).

جـ - يتم خزن بيانات الصورة لكل عنصر عرض (Pixel) نقوم بخزنها في مخزن وسطي (Buffer Image) وذلك لاستخدامه في المراحل اللاحقة . وجزء المعالجة البرمجي الخاص بهذه المرحلة هو :

```
for I:=0  to image1.picture.height-1 do
  for j:=0  to image1.picture.width-1 do
   begin
z1:=image1.canvas.pixels[i,j];
  for k:=1 to 24 do
     begin
       if z1 mod 2=0 then
          begin
            if k <= 8 then
                r[i,j]:=r[i,j]+'0'
            else if (k <= 16) then
            g[i,j]:=g[i,j]+'0'
          else   b[i,j]:=b[i,j]+'0';
         end
       else
          begin
               if k <= 8 then
 r[i,j]:=r[i,j]+'1'
 else if  (k<=16) then
 g[i,j]:=g[i,j]+'1'
else  b[i,j]:=b[i,j]+'1';
        end;
     z1:=z1 div 2;
   end;
   s1:=r[i,j]+g[i,j]+b[i,j];
   end;
```

يمكن تشفير صورة معتمداً على تحديد حواف أي لون في الصورة وتحديد عدد الألوان الخاصة بالصورة ،بعدها يتم تحليل القيم المستخرجة لتحديد الحواف لكي يتم تشفيرها بخوارزمية تبديل الأعمدة المستخدمة من خلال معامل (Laplacian operator) .

1) يمكن استخدام عدد من المفاتيح للتشفير مثلا (24) مفتاح من مفاتيح التشفير ، ويمكن فك التشفير للصورة ذاتها حيث يمكنها التعامل مع كل أنواع الصور ومختلف إحجامها إذ تقوم الإلية المعتمدة على تحويل أي نوع من الصور المراد تشفيرها أو فك التشفير إلى الصور ذات الامتداد (BMP).

عملية المعالجة الخاصة بالتشفير ويمكن توضيحها بالخطوات التالية :

أ- مرحلة المعالجة الابتدائية :

يتم في هذه الخطوة سحب بيانات كل عنصر عرض في الصورة وتحويله إلى مصفوفة ثنائية بأربعة صفوف وستة أعمدة (4*6) لانة قلنا أن عدد المفاتيح 24 . والجزء البرمجي الخاص بهذه الخطوة :

```
W:=1;
for f1:=1 to 4 do
   for f2:=1 to 6 do
   begin
     h1[f1,f2]:=s1[w];
     inc(w);
   end;
```

ب- مرحلة المعالجة الوسطية :

تعتبر أهم مرحلة في المعالجة،إن الطريقة المتبعة هي عملية إبدال ما بين أعمدة أو صفوف هذه المصفوفة وبالتالي الحصول على خلطة لونية جديدة مختلفة تماماً عن الخلطة اللونية الخاصة بالصورة الأصلية وبالنتيجة فانه لايمكن التعرف على هذه الصورة الجديدة .

يمكن توضيح عملية المعالجة هذه بالحالات الآتية :

1. إن المصفوفة الثنائية الخاصة بكل عنصر عرض في الصورة ذات حجم 4*6 فانه يمكن قراءتها بإحدى الحالتين:
   - قراءتها بصيغة صف _ عمود
   - قراءتها بصيغة عمود _ صف

وفي كلتا الحالتين فانه هناك نتيجة تشفير مختلفة للصورة المشفرة الناتجة .

2. إجراء عملية الإبدال (Swapping) وتوجد حالتين هما:
   - إبدال ما بين الأعمدة الخاصة بالمصفوفة
   - إبدال ما بين الصفوف الخاصة بالمصفوفة

وفي كلتا الحالتين فانه يوجد عملية تشفير مطبقه تختلف نتائجها وكفاءتها في تطبيق النظام ، والجزء البرمجي الخاص بهذين الخطوتين :

```
for f1:=1 to 4 do
   for f2:=1 to 6 do
       hc1[f1,f2]:=h1[f1,strtoint(edit1.text[f2])];
```

جـ - مرحلة المعالجة النهائية وإظهار النتائج :

عملية استرجاع البيانات بعد تشفيرها ووضعها في الصورة الجديدة المشفرة وتتم بالخطوات التالية :

1) إعادة مصفوفة العنصر الثنائية إلى مصفوفة أحادية والجزء البرمجي الخاص بها هو :

```
for f1:=1 to 4 do
for f2:=1 to 6 do
S2:=S2+hc1[f1,f2];
```

2) تحويلها من النظام الثنائي إلى النظام العشري والجزء البرمجي الخاص بها هو :

```
   for k:=1 to 24 do
       begin
         if s2[k]='1' then
         ss:=ss+p;
p:=2*p;
end;
```

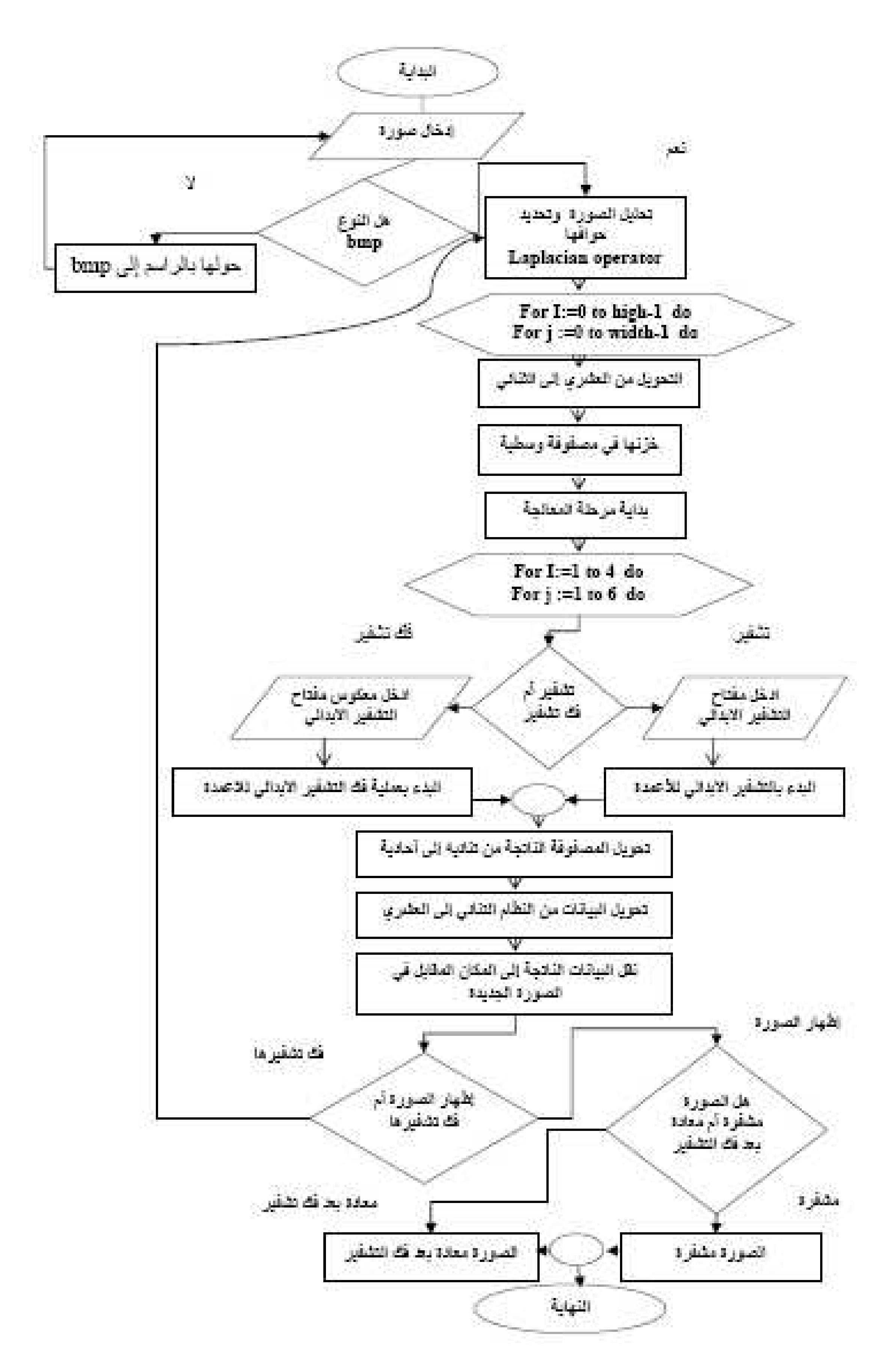
3) وضع هذه القيمة العشرية الجديدة لعنصر العرض في الجزء المخصص له في الصورة الجديدة(المشفرة) والجزء البرمجي الخاص به هو :

```
image2.canvas.pixels[i,j]:=ss;
```

إن مراحل تطبيق النظام التي تم توضيحها استخدمت في حالة إجراء عملية معالجة تشفير الصورة معينة بحجم معين لغرض إرسالها إلى الشخص معين عبر الانترنت أو لحمايتها من الانتشار بين المتطفلين أو السرقة.

إن مراحل عملية فك التشفير لبيانات هذه الصورة واستعادة بياناتها الأصلية (الصورة المعادة بعد فك التشفير) هي نفسها مراحل النظام المتمثلة في

- مرحلة تحليل الصورة
- مرحلة معالجة الصورة(المعالجة الأولية ، المعالجة الوسطية ،المعالجة النهائية)



شكل (94):المخطط الانسيابي لتشفير صورة رقمية

9-6 تشفير الفونيمات الصوتية التوقفية والاحتكاكية داخل صورة مشفرة:

إن تقسيم الفونيمات في اللغة العربية حسب طبيعة الأصوات المنتجة لها يكون معتمداً على طريقة نطقها من قبل المتكلم حيث تتأثر بطبيعة اللهجة الدارجة للمتكلم ،لذلك فان استخراج الهيكل الفونيمي والصوتي للغة غالباً ما يكون متأثر باللهجة المحلية .

ومن المعروف إن تقسم إلى قسمين رئيسيين هما فونيمات اللين (المتحركة)(Vowels) والفونيمات الساكنة (Constants) فالأولى تمتاز باحتوائها على منسوب طاقة صوتية عالي ويكون وضع المسار (المجرى الهوائي) الصوتي شبه مرتخي عند إخراج الأصوات ، أي تردداتها تكون واطئة في حين تمتاز الثانية بان منسوب الطاقة الصوتية قليل وتحصل انقباضات في مناطق مختلفة من المسار الصوتي عند الإخراج ،وبالتالي تكون ذات ترددات عالية والخصائص الصوتية تكون قليلة الوضوح والثبات ويختلف الوقت اللازم لإخراج الصوت الذي يسجل بالأمر الصوتي الزمني (Duration) والذي يؤثر فيه عدة عوامل أهمها سرعة الشخص المتكلم .

<u>نوعية الصورة والفونيمات الصوتية المستخدمة</u>

يتم تحويل الصورة المراد تطبيقها من أي نوع كانت إلى صورة نقطية (أي صورة ذات امتداد من نوع BMP .) وذلك لامكانيه التعامل المباشر مع محتويات هذا النوع من الصور حيث أنها لا تخضع لأي خوارزميات ضغط أو تشفير للبيانات الأصلية .

إما بالنسبة للملف الصوتي فقد استخدمت الفونيمات التوقفية (Phonemes of stopping) والفونيمات الاحتكاكية (Phonemes of attrition) للأصوات المستخدمة علماً إن الجدول التالي يوضح مواقع النطق الأفقية والعمودية الفونيمات المستخدمة من حيث جودة الصورة ونوع النطق .

جدول رقم (4): أنواع الفونيمات التوقفية والاحتكاكية الصوتية في اللغة العربية

مواقع النطـــق العمودي	مواقـع النطـــق الأفقيــــــــــة							نوعيـــة الأصوات	
	لهوي	حلقي	حنكي	نطقي	بيسني	شفويسني	شفوي		
	ق	ك		ط ت				غير صوتي	توقفـــي
			ج	د			ب	صوتي	
		خ	ش	ص س	ث ظ	ف		غير صوتي	احتكاكـــي
	غ			ز	ض ذ			صوتي	

الصور المطبقة لا تقتصر على حجم معين أو محدد حيث يمكن استعمال صورة بأي حجم كانت سواء صغيرة أو كبيرة ، إما بالنسبة لحجم الملف الصوتي فان الأمد الزمني (Duration) هو الذي يحدده حسب الكلمات المختارة المستخدمة والتي تحوي على الأصوات التوقفية والأصوات الاحتكاكية .

ونستطيع تلخيصها بالعمليات الآتية :

أ- تحليل الصورة وإجراء عملية مسح (SCANNING) لها للحصول على بيانات كل عنصر عرض في الصورة (Pixel) وعادة ما تكون موضوعه بالنظام العشري (Decimal) .

ب- عملية تحويل صيغة بيانات الصورة من النظام العشري (Decimal System) إلى النظام الثنائي (Binary System) حيث إن كل عنصر (Pixel) في الصورة يتم تحويله إلى 24 بت في النظام الثنائي . وبما إن كل عنصر عرض يتكون من ثلاث ألوان أساسية مشتركة في تكوينه وهي الأحمر،الأخضر،الأزرق (نظام ألوان RGB) ، فان لكل لون من هذه الألوان يُحجز 8 بتأت خاصة به وبالنتيجة فان الشدة اللونية لكل لون سوف تتراوح ما بين (0_ 255).

جـ - بعد الحصول على بيانات الصورة نقوم بخزنها في مخزن وسطي (Buffer Image) وذلك لاستخدامه في المراحل اللاحقة . وجزء المعالجة البرمجي الخاص بهذه المرحلة هو :

```
for I:=0  to image1.picture.height-1 do
  for j:=0  to image1.picture.width-1 do
   begin
z1:=image1.canvas.pixels[i,j];
  for k:=1 to 24 do
     begin
       if z1 mod 2=0 then
          begin
            if k <= 8 then
               r[i,j]:=r[i,j]+'0'
            else if (k <= 16) then
                  g[i,j]:=g[i,j]+'0'
                else   b[i,j]:=b[i,j]+'0';
          end
       else
          begin
                   if k <= 8 then
              r[i,j]:=r[i,j]+'1'
            else if  (k<=16) then
                  g[i,j]:=g[i,j]+'1'
                else  b[i,j]:=b[i,j]+'1';
          end;
     z1:=z1 div 2;
   end;
   s1:=r[i,j]+g[i,j]+b[i,j];
   end;
```

مثال//

حيث يتم تسجيل أصوات (16) متكلم (8) Male و (8) Female وبأعمار مختلفة ،لكل متكلم 8 كلمات ينطقها تحتوي على الفونيمات الاحتكاكية والتوقفية والجدول (5) يبين الكلمات المنطوقة وتفاصيلها .

جدول(5):الكلمات المحتوية على الفونيمات الاحتكاكية والتوقفية

الفهرس	الكلمة المنطوقة	الملاحظات
1	جدك	تحتوي على صوتين توقفين مسموعين وصوت توقفي غير مسموع
2	قط	تحتوي على صوتين توقفين غير مسموعين Un voice stop sounds
3	جد	تحتوي على صوتين توقفين مسموعين Voiced stop sounds
4	شخص	ثلاثة أصوات احتكاكية غير مسموعة
5	صنع	صوتين احتكاكين مسموعين
6	فض	صوت احتكاكي غير مسموع وصوت احتكاكي مسموع
7	قفص	صوت توقفي غير مسموع وصوتين احتكاكين مسموعين
8	سكب	صوت احتكاكي غير مسموع وصوت توقفي غير مسموع وصوت توقفي مسموع

حيث يتم سحب بيانات الملف الصوتي من خلال المعاملات الخاصة بالصوت بعد تسجيله وإضافته إلى بيانات الصورة الأصلية بعد تحويله إلى النظام الثنائي(Binary system) أيضا حيث يتم دمج بيانات الملف الصوتي في نهاية الملف الخاص ببيانات الصورة ، والجدول(6) التالي يوضح قيم معاملات الكلمة(قفص) ينطقها اثنين من المتكلمين مع معاملاتها .

الجدول(6) التالي يوضح قيم معاملات الكلمة(قفص) ينطقها اثنين من المتكلمين مع معاملاتها .

المعاملات	اسم الفايل الصوتي	
	HA	HI
عدد الكتل	63	68
الأمد الزمني	1.5587301587314	1.6837188208714
التقاطع الصفري	567.259489414091	5.24296675192747
اكبر تقاطع صفري	0.15217391309199	0.134057971014511
معدل التردد الأساسي	203.796046223512	171.654874090915
اكبر تردد أساسي	538.168002302869	438.90161801911
الطاقة	117.22657992891	40.464424903181
الوسع	3.842195215932	2.18937031118986

والمعاملات المستخرجة من الملف الصوتي هي :

1. **نسبة الطاقة للزمن القصير:**

تمثل طاقة الإشارة الصوتية التغير في وسع الصوت (Amplitude) الذي يعتبر من العوامل المهمة التى توضح خواص الأصوات المنطوقة للمتكلم ، والمعادلة التالية توضح ذلك :

$$En = \sum_{m}^{N-1} \left[X(m).W(n+m) \right]^2 \quad ....(66)$$

عندما :

N= تمثل إطار العينات

n= تمثل عدد الإطارات

X(m) = تمثل إشارة الصوت لموقع (m)

W(m) = تمثل النافذة المستخدمة

عدد العينات يتراوح من (100-200) عينة لكل إطار مع قسمة وقت تتراوح بين (10-20) ملي ثانية.

2. **نسبة التقاطع الصفري للزمن القصير:**

والذي يحدث للتغير في الإشارة الصوتية خلال موجه الصوت للمحور الزمني الصوتي (يحدث التقاطع الصفري للإشارة في كل مرة تعدي الموجة لمحور الزمن)، والمعادلة التالية توضح ذلك:

$$Zn = \left| \mathrm{sgn}[X(m) - \mathrm{sgn}[X(m-1)\right| w(n-m) \quad ....(67)$$

Sgn(X(n))= 1 for X(n)>0 …………………...(68)

-1 for X(n)<0

عندما :

التقاطع الصفري يحدث بين (n-n-1).

[X(n)] # sing[X(n-1)] ....(69)

فإذا كانت إشارة الصوت تمتلك تردد أساسي [F0] ومعدل عينات [Fs] فان :

Zcr = 2F0/Fs ....(70)

F0 = (Zcr * Fs) /2 ………….(71)

3. **معدل وسع الزمن القصير :**

تعكس طاقة الزمن القصير تغيرات الوسع للصوت وتعطي الخواص المعتمدة للمتكلم بصورة جيدة وتنتج التغيرات في طاقة الكلام بواسطة التغيرات في الضغط ألمزماري الجزئي وشكل المسار الصوتي والمعادلة التالية توضح ذلك:

$$Mn=\sum_{m}^{N-1} X(m)*W(n+m)$$ ....(72).

إما النافذة المستخدمة في تحليل الصوت هي (Hamming Window):

$$W(n)=\begin{cases} 0.5-0.46\cos(2\pi n/(N-1)) & 0\le n\le N-1 \\ 0 & otherwise \end{cases}$$ ....(73)

مرحلة تشفير الملف الصوتي والصورة:

تتم بالمرحل التالية :

أ- مرحلة المعالجة الابتدائية : يتم في هذه الخطوة سحب بيانات كل عنصر عرض في الصورة وتحويله إلى

مصفوفة ثنائية بأربعة صفوف وستة أعمدة (4*6) والجزء البرمجي الخاص بهذه الخطوة :

```
W:=1;
for f1:=1 to 4 do
   for f2:=1 to 6 do
   begin
     h1[f1,f2]:=s1[w];
     inc(w);
   end;
```

ب-مرحلة المعالجة الوسطية :

إن كفاءة النظام ونجاحه تعتمد على هذه المرحلة وقدرتها على تشفير بيانات الصورة ، حيث تتم عملية إبدال ما بين أعمدة أو صفوف هذه المصفوفة وبالتالي الحصول على خلطة لونية تختلف عما كانت علية في الصورة الأصلية . وبالنتيجة فانه لايمكن التعرف على هذه الصورة الجديدة ، يمكن توضيح عملية المعالجة هذه بالحالات الآتية :

1. تقرأ المصفوفة :
   - بصيغة صف _ عمود
   - بصيغة عمود _ صف

2. إجراء عملية الإبدال (Swapping) :
   - إبدال ما بين الأعمدة الخاصة بالمصفوفة
   - إبدال ما بين الصفوف الخاصة بالمصفوفة

وفي كلتا الحالتين فانه يوجد عملية تشفير مطبقه تختلف نتائجها وكفاءتها في تطبيق النظام ، والجزء البرمجي الخاص بهذين الخطوتين :

```
for f1:=1 to 4 do
    for f2:=1 to 6 do
      hc1[f1,f2]:=h1[f1,strtoint(edit1.text[f2])];
```

جـ - مرحلة المعالجة النهائية وإظهار النتائج :

عملية استرجاع البيانات بعد تشفيرها ووضعها في الصورة الجديدة المشفرة وتتم بالخطوات التالية :

1) إعادة مصفوفة العنصر الثنائية إلى مصفوفة أحادية والجزء البرمجي الخاص بها هو :

```
for f1:=1 to 4 do
  for f2:=1 to 6 do
      S2:=S2+hc1[f1,f2];
```

2) تحويلها من النظام الثنائي إلى النظام العشري والجزء البرمجي الخاص بها هو :

```
for k:=1 to 24 do
    begin
      if s2[k]='1' then
       ss:=ss+p;
       p:=2*p;
    end;
```

3) وضع هذه القيمة العشرية الجديدة لعنصر العرض في الجزء المخصص له في الصورة الجديدة(المشفرة) ، والجزء البرمجي الخاص به هو :

```
image2.canvas.pixels[i,j]:=ss;
```

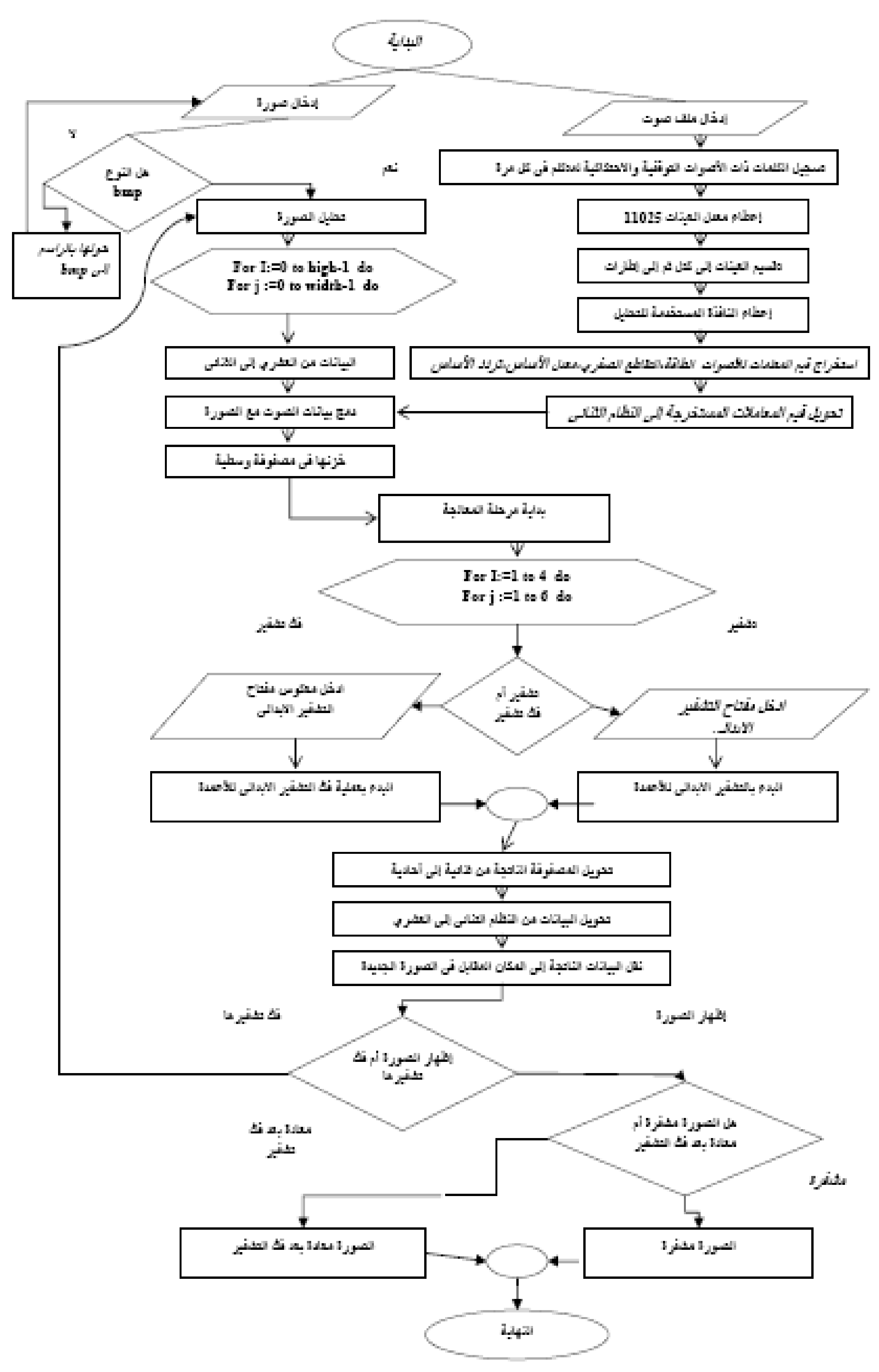
مرحلة فك تشفير الملف الصوتي والصورة:

تتمثل بالعمليات التالية :

- عملية تحليل الصورة
- عملية معالجة الصورة(المعالجة الأولية ، المعالجة الوسطية ، المعالجة النهائية)

مع استخدام معكوس المفتاح الابدالي الخاص بالتشفير الذي سيستخدم لفك التشفير والذي سنوضحه لاحقا .

المخطط الانسيابي (95) التالي يوضح خطوات تطبيق النظام



شكل رقم (95) المخطط الانسيابي لتوضيح عمل تشفير ملف صوتي داخل صورة مشفرة

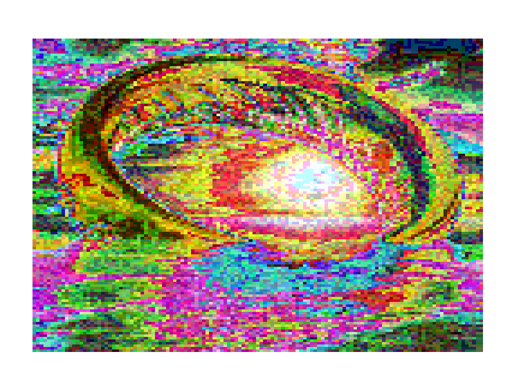
والشكل الاتي يوضح عملية التشفير وفك التشفير للنظام :





Cipher image
The key: 261534

Decipher image
The inverse of key:315642





Cipher image
The key: 416235

Decipher image
The inverse of key:245163

شكل رقم (96) يوضح عملية التشفير لملف صوتي داخل صورة بحجم (128*128) وباستخدام عدة مفاتيح

جدول(4) يوضح القيم العشرية لعناصر الصورة اللونية لمقطع حجمه (10*10) من الصورة ألمستخدمه في مثال التطبيق .

pixels	Pix0	Pix1	Pix2	Pix3	Pix4	Pix5	Pix6	Pix7	Pix8	Pix9
Pix0	14929332	14929332	14929078	15060664	14994871	14863285	15060405	15126198	14994867	14994867
Pix1	14929587	14863539	14929078	14994871	14929078	14797492	15126198	14928819	14863281	15060660
Pix2	14929332	14863539	14929078	14994871	14994871	14929078	15060405	14928819	14994867	15126453
Pix3	14929332	14929332	14994871	15126457	15126457	15126711	14863026	15060405	15258039	15192246
Pix4	14863539	14929332	14994871	15060664	15126711	15126711	14928819	14994867	15192246	15258039
Pix5	14863539	14929332	14994871	14994871	14929332	14929332	15126453	14797488	14797488	15192759
Pix6	14863539	14929332	14995125	14929332	14863539	14863539	15192246	14929074	14732208	15061173
Pix7	14863539	14995125	15060918	14995125	14929332	14929332	15060660	15192759	15061173	14864308
Pix8	14797746	14995125	14863539	14995125	14995380	15192759	15061173	15192759	15324345	14930101
Pix9	14863539	14995125	14863539	14929332	14929587	15061173	15061173	14929587	15127480	14733237

جدول(5) يوضح القيم الثنائية المناظرة للقيم العشرية لعناصر الصورة اللونية في الجدول أعلاه .

pixel	Pix0	Pix1	Pix2	Pix3	Pix4	Pix5	Pix6	Pix7	Pix8	Pix9
Pix0	001011 011011 001111 000111	001011 011011 001111 000111	011011 010011 001111 000111	000111 010111 001110 100111	111011 011011 001100 100111	101011 011101 001101 000111	101011 011011 001110 100111	011011 010111 001101 100111	110011 011011 001100 100111	110011 011011 001100 100111
Pix1	110011 010111 001111 000111	110011 010011 001101 000111	011011 010011 001111 000111	111011 011011 001100 100111	011011 010011 001111 000111	001011 010101 001110 000111	011011 010111 001101 100111	110011 011101 001111 000111	100011 011101 001101 000111	001011 010111 001110 100111
Pix2	001011 011011 001111 000111	110011 010011 001101 000111	011011 010011 001111 000111	111011 011011 001100 100111	111011 011011 001100 100111	011011 010011 001111 000111	101011 011011 001110 100111	110011 011101 001111 000111	110011 011011 001100 100111	101011 011111 001101 100111
Pix3	001011 011011 001111 000111	001011 011011 001111 000111	111011 011011 001100 100111	100111 011111 001101 100111	100111 011111 001101 100111	111011 010000 101101 100111	010011 010101 001101 000111	101011 011011 001110 100111	111011 011000 101100 010111	011011 010000 101111 100111
Pix4	110011 010011 001101 000111	001011 011011 001111 000111	111011 011011 001100 100111	000111 010111 001110 100111	111011 010000 101101 100111	111011 010000 101101 100111	110011 011101 001111 000111	110011 011011 001100 100111	011011 010000 101111 100111	111011 011000 101100 010111
Pix5	110011 010011 001101 000111	001011 011011 001111 000111	111011 011011 001100 100111	111011 011011 001100 100111	001011 011011 001111 000111	001011 011011 001111 000111	101011 011111 001101 100111	000011 010101 001110 000111	000011 010101 001110 000111	111011 010100 101111 100111
Pix6	110011 010011 001101 000111	001011 011011 001111 000111	101011 010111 001100 100111	001011 011011 001111 000111	110011 010011 001101 000111	110011 010011 001101 000111	011011 010000 101111 100111	010011 010011 001111 000111	000011 011101 001100 000111	101011 010000 101110 100111
Pix7	110011 010011 001101 000111	101011 010111 001100 100111	011011 011111 001110 100111	101011 010111 001100 100111	001011 011011 001111 000111	001011 011011 001111 000111	001011 010111 001110 100111	111011 010100 101111 100111	101011 010000 101110 100111	001011 011111 001101 000111
Pix8	010011 011101 001110 000111	101011 010111 001100 100111	110011 010011 001101 000111	101011 010111 001100 100111	001011 011111 001100 100111	111011 010100 101111 100111	101011 010000 101110 100111	111011 010100 101111 100111	100111 010010 101110 010111	101011 010000 101111 000111
Pix9	110011 010011 001101 000111	101011 010111 001100 100111	110011 010011 001101 000111	001011 011011 001111 000111	110011 010111 001111 000111	101011 010000 101110 100111	101011 010000 101110 100111	110011 010111 001111 000111	000111 011100 101101 100111	101011 011111 001100 000111

الملاحق

جميـــــع البرامـج الخاصــة بمعالجة الصورة الرقمية

برنامج استخدام فلتر التحسين

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/*********************************/
FILE  *fp,*fn,*fd;
/********************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
int mask[9]={0,1,0,1,5,1,0,1,0};
unsigned char color[256][3];
bmphead bmp;
///////////////////////////////
int detectsvga(void)
{return 0;
```

```
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
```

```
for(j=0;j<=bmp.width-1;j++)
{gr[i][j]=bmp_line[j];
putpixel(j,i,bmp_line[j]);
}
lw=lw-nw;
}
int ii=0;int l;
int s=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{
for(int k=i;k<i+3;k++)
for(int l=j;l<j+3;l++)
{if(ii==1||ii==3||ii==5||ii==7)
s=s+gr[k][l]*(0-mask[ii]);
else
s=s+gr[k][l]*mask[ii];
ii++;
}
a[i+1][j+1]=s;
s=0;
ii=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+100,i,a[i][j]);
fclose(fp);
//histogram();
getchar();
getchar();
getchar();
closegraph();
}
//return(0);
/////////////////////////////
```

برنامج لاستخدام العمليات التالية:

1-cut[1]
2-cut[2]
3-cut[3]
4-cut[4]
5-zoero order zooming
6-average zooming
7-first convolution
8-general zomming
9-add
10-sub
11-mult
12-div
13-and
14-or
15-not

Sol:

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
```

```
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd,*ff;
/*******************************/
int compute(int);
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],gr1[100][100];
unsigned char far a[100][100];
unsigned char far a1[100][200],a2[200][200];
unsigned char far n1[100][100];
unsigned char color[256][3];
  unsigned char far b1[200][200];
     unsigned char far b2[205][205];
    unsigned char far b3[200][200];
float mask1[9]={0.25,0.5,0.25,0.5,1,0.5,0.25,0.5,0.25};
bmphead bmp;
float mask[3][3];
/*int k=0;
for(int ii=0;ii<3;ii++)
{for(int jj=0;jj<3;jj++)
mask[ii][jj]=mask1[k];k++;}*/
//////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
int compute(int x1)
{int x[9];
for(i=0;i<8;i++)
{x[i]=x1%2;
```

```
x1=x1/2;
}
for(i=0;i<8;i++)
{if(x[i]==0)
   x[i]=1 ;
if (x[i]==1)
  x[i]=0;
 }
 int s=0;
 int j=7;
for(i=0;i<8;i++)
{int l=pow(2,j);
s=s+(l*x[i]);
j--;
}
return s;
}
//////////////////////////////
void main()
{clrscr();
int no;
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25],file_name1[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
```

```
    printf("enter file name to be loaded >>   ");
scanf("%s",file_name1);
    cout<<"##########"<<"\n";
    cout<<"**********"<<"\n";
    cout<<"1-cut[1]"<<"\n";
    cout<<"2-cut[2]"<<"\n";
    cout<<"3-cut[3]"<<"\n";
    cout<<"4-cut[4]"<<"\n";
    cout<<"5-zoero order zooming"<<"\n";
    cout<<"6-average zooming"<<"\n";
    cout<<"7-first convolution"<<"\n";
    cout<<"8-general zomming"<<"\n";
    cout<<"9-add"<<"\n";
    cout<<"10-sub"<<"\n";
    cout<<"11-mult"<<"\n";
    cout<<"12-div"<<"\n";
    cout<<"13-and"<<"\n";
    cout<<"14-or"<<"\n";
    cout<<"15-not"<<"\n";
    cout<<"***********************"<<"\n";
    cout<<"#######################"<<"\n";
    cout<<"****inter yout selection****  ";
    cin>>no;
/////////////array/of/image////////////////////
void d();
{int r=99;
/*unsigned char far*/
cout<<r;
}
long int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
getchar();
```

```
switch(no)
{ case 1:
  for(i=0;i<50;i++)
  for(j=0;j<50;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   ////////////////////////////
   case 2:
  for(i=0;i<50;i++)
  for(j=50;j<=100;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   /////////////////////////
   case 3:
  for(i=50;i<=100;i++)
  for(j=0;j<50;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   ////////////////////////
   case 4:
  for(i=50;i<=100;i++)
  for(j=50;j<=100;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   /////////////////////////
   case 5:
   {for(i=0;i<=bmp.depth;i++)
    {x1=0;
    for(j=0;j<=bmp.width;j++)
    {a1[i][x1]=gr[i][j];
     x1++;
     a1[i][x1]=gr[i][j];
     x1++;
     }}
     for(i=0;i<=2*bmp.depth;i++)
     {y1=0;
     for(j=0;j<=bmp.width;j++)
     {a2[y1][i]=a1[j][i];
     y1++;
```

```
 a2[y1][i]=a1[j][i];
 y1++;}}
 for(i=0;i<=2*bmp.depth;i++)
 for(j=0;j<=2*bmp.width;j++)
 putpixel(j+101,i,a2[i][j]);
 }break;
 /////////////////////////////
case 6:
{for(i=0;i<=bmp.depth;i++)
 {x1=0;
 for(j=0;j<=bmp.width;j++)
 {a1[i][x1]=gr[i][j];
  x1++;
  a1[i][x1]=(gr[i][j]+gr[i][j+1])/2;
  x1++;
  }}
  for(i=0;i<=2*bmp.depth;i++)
  {y1=0;
  for(j=0;j<=bmp.width;j++)
  {a2[y1][i]=a1[j][i];
  y1++;
  a2[y1][i]=(a1[j][i]+a1[j+1][i])/2;
  y1++;}}
  for(i=0;i<=2*bmp.depth;i++)
  for(j=0;j<=2*bmp.width;j++)
  putpixel(j+101,i,a2[i][j]);
  }break;
  ///////////////////////
  case 7:
  {int ii=0;

  for(i=0;i<100;i++)
  {for(j=0;j<100;j++)
  b1[ii][j]=0;
  ii++;
  for(j=0;j<100;j++)
  b1[ii][j]=gr[i][j];
  ii++;
  }
  for(j=0;j<100;j++)
  b1[ii][j];
  int jj=0;
  for(j=0;j<100;j++)
```

```
{for(i=0;i<201;i++)
b2[i][jj]=0;
jj++;
for(j=0;j<201;j++)
b2[i][jj]=b1[i][j];
jj++;
}
ii=0;
int s=0;int k,l;
for(i=0;i<199;i++)
for(j=0;j<199;j++)
{for(k=i;k<i+3;k++)
for(l=j;l<j+3;l++)
{s=s+(b2[k][l]*mask1[ii]);
i++;
}
b3[i][j]=s;
ii=0; s=0;
}
for(i=0;i<199;i++)
for(j=0;j<199;j++)
putpixel(j+150,i,b3[i][j]);
    }break;
    //////////////////////
    case 15:
    {for(i=0;i<100;i++)
     for(j=0;j<100;j++)
     {int l=compute(gr[i][j]);
      n1[i][j]=l;
      putpixel(j+100,i,n1[i][j]);
      }
      }}
    fclose(fp);
 ff=fopen(file_name1,"rb");
fread(&bmp,1,sizeof(bmphead),ff);

 for(i=0;i< 255;++i)
  {c1=fgetc(ff);
   c1=c1>>2;
   color[i][0]=c1;
   c2=fgetc(ff);
   c2=c2>>2;
   color[i][1]=c2;
```

```
    c3=fgetc(ff);
    c3=c3>>2;
    color[i][2]=c3;
    fgetc(ff);
    }
    nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(ff,lw,0);
  fread(bmp_line,nw,1,ff);
  for(j=0;j<=bmp.width-1;j++)
  {a[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  //histogram();
  getchar();
  getchar();
  closegraph();
  }
  //return(0);
```

برنامج لاستخدام وتحليل قيم الصورة histogram :

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
```

```
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/*******************************/
FILE  *fp,*fn,*fd,*ff;
/******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],gr1[100][100];
unsigned char far gr2[100][100];
unsigned char color[256][3];
bmphead bmp;
///////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
///////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4,c;
char bmp_line[1024],file_name[25],file_name1[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
```

```
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);
  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
     printf("enter anather file name to be loaded >>     ");
scanf("%s",file_name1);
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
  }
  fclose(fp);
////////////array/of/image////////////////////

  getchar();
  getchar();
  ff=fopen(file_name1,"rb");
fread(&bmp,1,sizeof(bmphead),ff);
  for(i=0;i< 255;++i)
    {c1=fgetc(ff);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(ff);
     c2=c2>>2;
```

```
    color[i][1]=c2;
    c3=fgetc(ff);
    c3=c3>>2;
    color[i][2]=c3;
    fgetc(ff);
    }
    nw=4*((bmp.width+3)/4);
    lw=(bmp.depth-1)*nw+1078;
    for(i=0;i<=bmp.depth-1;i++)
     { fseek(ff,lw,0);
      fread(bmp_line,nw,1,ff);
      for(j=0;j<=bmp.width-1;j++)
      {gr1[i][j]=bmp_line[j];
      putpixel(j+100,i,bmp_line[j]);
      }
      lw=lw-nw;
      }
          fclose(ff);
  //histogram();
  getchar();
  getchar();
  getchar();
  closegraph();
  }
  //return(0);
  /////////////////////////////
```

برنامج لايجاد فلتر الوسيط madian

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
```

```
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
unsigned char ss[9];
unsigned char color[256][3];
bmphead bmp;
////////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
///////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
```

```
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
/////////////array/of/image///////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
   }
  lw=lw-nw;
}
  int l,s;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {int x=0;
  for(int k=i;k<i+3;k++)
  for(int l=j;l<j+3;l++)
  {ss[x]=gr[k][l];
  x++;
   }
```

```
for(int t1=0;t1<8;t1++)
{for(int t2=t1+1;t2<9;t2++)
{if(ss[t1]>ss[t2])
{int o=ss[t1];
ss[t1]=ss[t2];
ss[t2]=o;
}}}
s=ss[4];
a[i][j]=s;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
fclose(fp);
//histogram();
getchar();
getchar();
getchar();
closegraph();
}
//return(0);
/////////////////////////////
```

برنامج لايجاد Recorce

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
```

```
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/*******************************/
FILE  *fp,*fn,*fd;
/******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
unsigned char color[256][3];
bmphead bmp;
//////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
```

```
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
void d();
{int r=99;
cout<<r;
}
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
   }
  lw=lw-nw;
}
  fclose(fp);
  //histogram();
  getchar();
  getchar();
  getchar();
  closegraph();
   }
  //return(0);
  ////////////////////////////
```

برنامج لاستخدام Robert filter

```
#include<math.h>
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
int z[4];
unsigned char color[256][3];
bmphead bmp;
////////////////////////////////
```

```
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
```

```
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}

  int l;
  int ss=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {int x=0;
  for(int k=i;k<i+2;k++)
  for(int l=j;l<j+2;l++)
  {z[x]=gr[k][l];
  x++;
  }
  int s1=(z[2]-z[1])*(z[2]-z[1]);
  int s2=(z[3]-z[0])*(z[3]-z[0]);
  ss=sqrt(s1+s2);
  a[i+1][j+1]=ss;
  ss=0;
  }
  for(i=0;i<100;i++)
  for(j=0;j<100;j++)
  putpixel(j+110,i,a[i][j]);
  getchar();
  getchar();
  getchar();
  fclose(fp);
  //histogram();
  getchar();
  getchar();
  getchar();
  closegraph();
  }
  //return(0);

  /////////////////////////////
```

برنامج لاستخدام Robinson filter :

```
#include<iostream.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/********************************/
int q,i=0,j=0,m,n,x1,y1;
int gdriver,gmode;
int s;
unsigned char far gr[100][100],a[100][100];
 int mask0[9]={-1,0,1,-2,0,2,-1,0,1};
 int mask1[9]={0,1,2,-1,0,1,-2,-1,0};
 int mask2[9]={1,2,1,0,0,0,-1,-2,-1};
 int mask3[9]={2,1,0,1,0,-1,0,-1,-2};
 int mask4[9]={1,0,-1,2,0,-2,1,0,-1};
```

```
 int mask5[9]={0,-1,-2,1,0,-1,2,1,0};
 int mask6[9]={-1,-2,-1,0,0,0,1,2,1};
 int mask7[9]={-2,-1,0,-1,0,1,0,1,2};
 int max[8];
unsigned char color[256][3];
bmphead bmp;
///////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
```

```
    }
////////////array/of/image//////////////////
int x1,y1;
int s0,s1,s2,s3,s4,s5,s6,s7;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  s0=0;s1=0;s2=0;s3=0;s4=0;s5=0;s6=0;s7=0;
  int z=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {
  for(int k=i;k<i+3;k++)
  for(int l=j;l<j+3;l++)
  {
  s0=s0+(gr[k][l]*mask0[z]);
  s1=s1+(gr[k][l]*mask1[z]);
  s2=s2+(gr[k][l]*mask2[z]);
  s3=s3+(gr[k][l]*mask3[z]);
  s4=s4+(gr[k][l]*mask4[z]);
  s5=s5+(gr[k][l]*mask5[z]);
  s6=s6+(gr[k][l]*mask6[z]);
  s7=s7+(gr[k][l]*mask7[z]);
  z++;
  }
  max[0]=s0;
  max[1]=s1;
  max[2]=s2;
  max[3]=s3;
  max[4]=s4;
  max[5]=s5;
  max[6]=s6;
  max[7]=s7;
  x=0;
```

```
 for(int t=0;t<8;t++)
 {if(max[t]>=x)
 x=max[t];}
 a[i+1][j+1]=x;
   z=0;
  s0=0; s1=0;
  s2=0; s3=0; s4=0; s5=0;
  s6=0; s7=0;
 }
 for(i=0;i<100;i++)
 for(j=0;j<100;j++)
 putpixel(j+110,i,a[i][j]);
 fclose(fp);
 //histogram();
 getchar();
 getchar();
 getchar();
 closegraph();
 }
 //return(0);
 //////////////////////////////
```

برنامج لاستخدام الغمليات المنطقية وفلاتر الترشيح

```
#include<math.h>
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
```

```
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,l,t,z,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],gr1[1600],gr3[12800];
unsigned char far   g1[100],gr4[200][200],g2[255];
unsigned char far a[100][100];
unsigned char far  b1[100][100];
unsigned char far  x2[10];
unsigned char far  gr2[100];
unsigned char far  g3[100];
unsigned char far  z1[4];
 int f0[9]={-3,-3,5,-3,0,5,-3,-3,5};
 int f1[9]={-3,5,5,-3,0,5,-3,-3,-3};
 int f2[9]={5,5,5,-3,0,-3,-3,-3,-3};
 int f3[9]={5,5,-3,5,0,-3,-3,-3,-3};
 int f4[9]={5,-3,-3,5,0,-3,5,-3,-3};
 int f5[9]={-3,-3,-3,5,0,-3,5,5,-3};
 int f6[9]={-3,-3,-3,-3,0,-3,5,5,5};
 int f7[9]={-3,-3,-3,-3,0,5,-3,5,5};
unsigned char ss[9];
float mask1[9]={0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};
int mask[9]={0,1,0,1,5,1,0,1,0};
int mask11[9]={(0,-2,-1),(0,0,0),(1,2,1)};
int mask12[9]={(-1,-2,-1),(0,0,0),(1,2,1)};
int n1[9]={(-1,-1,-1),(0,0,0),(1,1,1)};
int n2[9]={(-1,-1,-1),(0,0,0),(1,1,1)};
int l7[9]={-2,-1,0,-1,0,-1,0,1,2};
```

```
int l0[9]={-1,0,1,-2,0,2,-1,0,1};
int l1[9]={0,1,2,-1,0,1,-2,-1,0};
int l2[9]={1,2,1,0,0,0,-1,-2,-1};
int l3[9]={2,1,0,1,0,-1,0,-1,-2};
int l4[9]={1,0,-1,2,0,-2,1,0,-1};
int l5[9]={0,-1,-2,1,0,-1,2,1,0};
int l6[9]={-1,-2,-1,0,0,0,1,2,1};
int max[8];
unsigned char color[256][3];
bmphead bmp;
///////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)

setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
///////////////////////////////
 void pro();

void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
```

```
    c2=fgetc(fp);
    c2=c2>>2;
    color[i][1]=c2;
    c3=fgetc(fp);
    c3=c3>>2;
    color[i][2]=c3;
    fgetc(fp);
    }
    cout<<"\n";
    cout<<"*1..or"<<"\n"<<"\n";
    cout<<"*2..not"<<"\n"<<"\n";
    cout<<"*3..and"<<"\n"<<"\n";
    cout<<"*4..meanfilter"<<"\n"<<"\n";
    cout<<"*5..medainfilter"<<"\n";
    cout<<"*6..enhancment"<<"\n";
    cout<<"*7..robert"<<"\n";
    cout<<"*8..sobel"<<"\n";
    cout<<"*9..prewit"<<"\n";
    cout<<"*10..robinson"<<"\n";
    cout<<"*11..kirsch"<<"\n";
    cout<<"*12..histogram" <<"\n";
    cout<<"13*..proparity"<<"\n";
    cout<<"13*..standard deviation\n";
    cout<<"14:mean*********** enter your
select***************"<<"\n";
    int no;
    cin>>no;

////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
 putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  fclose(fp);
```

```
getchar();
getchar();
//////////////////////
 switch (no)
 {case 1:
 {
for(int i=0;i<100;i++)
for(j=0;j<100;j++)
  a[i][j]=254;
 for(i=0;i<30;i++)
 for(j=0;j<30;j++)
 a[i][j]=0;
 for(i=0;i<100;i++)
 for(j=0;j<100;j++)
 putpixel(j+100,i,a[i][j]);
for(i=0;i<100;i++)
for(j=0;j<100;j++)
{int t=gr[i][j];
int t1=a[i][j];
  for(int i2=7;i2>=0;i2--)
  {g1[i2]=t%2;
   t=t/2;
   gr2[i2]=t1%2;
   t1=t1/2;
   }
 for(int l=0;l<8;l++)
  gr3[l]=g1[l]|gr2[l];
int c=2; z=7;l=0;int i1=0;
 for(int j1=0;j1<8;j1++)
 {int t2=pow(c,z);z=z-1;
 x2[j1]=t2*gr3[i1];
 l=l+x2[j1];i1=i1+1;
 }
 if (l>254)
   l=254;
b1[i][j]= l;
}
getchar();
  for(i=0;i<100;i++)
  for(j=0;j<100;j++)
    putpixel(j+110,i+110,b1[i][j]);
getchar();
getchar();
```

```
}break;
//////////////////////////////////////
case 2:
{ for(i=0;i<bmp.depth;i++)
  for(j=0;j<bmp.width;j++)
{   int t=gr[i][j];
  for(int i2=7;i2>=0;i2--)
  {gr2[i2]=t%2;
   t=t/2;
   }
 for(int l=0;l<8;l++)
 { if(gr2[l]==0)
 gr2[l]=1 ;
 else
 gr2[l]=0;
 }
   int c=2;
 int z=7;l=0;int i1=0;
 for(int j1=0;j1<8;j1++)
 {int t2=pow(c,z);z=z-1;
 gr1[j1]=t2*gr2[i1];
 l=l+gr1[j1];i1=i1+1;
 }
 if (l>255)
   l=255;
gr4[i][j]=l;
}
for(i=0;i<bmp.depth;i++)
for(j=0;j<bmp.width;j++)
  putpixel(j+100,i+100,gr4[i][j]);
   }
  getchar();
  getchar();
  getchar();
  break;
//////////////////////////////////////
  case 3:
   {for(int i=0;i<100;i++)
for(j=0;j<100;j++)
  a[i][j]=0;
 for(i=0;i<30;i++)
 for(j=0;j<30;j++)
 a[i][j]=254;
```

```
 for(i=0;i<100;i++)
 for(j=0;j<100;j++)
 putpixel(j+100,i,a[i][j]);
for(i=0;i<100;i++)
for(j=0;j<100;j++)
{int t=gr[i][j];
int t1=a[i][j];
  for(int i2=7;i2>=0;i2--)
  {g1[i2]=t%2;
   t=t/2;
   gr2[i2]=t1%2;
   t1=t1/2;
   }
 for(int l=0;l<8;l++)
  gr3[l]=g1[l]&gr2[l];
int c=2; z=7;l=0;int i1=0;
 for(int j1=0;j1<8;j1++)
 {int t2=pow(c,z);z=z-1;
 x2[j1]=t2*gr3[i1];
 l=l+x2[j1];i1=i1+1;
 }
 if (l>254)
   l=254;
b1[i][j]= l;
}
  for(i=0;i<100;i++)
  for(j=0;j<100;j++)
    putpixel(j+100,i+100,b1[i][j]);
   getchar();
   getchar();
   getchar();
  }break;
  ////////////////////////////////////////////
  case 4:
  { int ii=0;int l;
  int s=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {
  for(int k=i;k<i+3;k++)
  for(int l=j;l<j+3;l++)
  {
  s=s+gr[k][l]*mask1[ii];
```

```
ii++;
}
a[i+1][j+1]=s;
s=0;
ii=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+100,i,a[i][j]);
}break;
///////////////////////////////
case 5:
{  int l,s;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{int x=0;
for(int k=i;k<i+3;k++)
for(int l=j;l<j+3;l++)
{ss[x]=gr[k][l];
x++;
}
for(int t1=0;t1<8;t1++)
{for(int t2=t1+1;t2<9;t2++)
{if(ss[t1]>ss[t2])
{int o=ss[t1];
ss[t1]=ss[t2];
ss[t2]=o;
}}}
s=ss[4];
a[i][j]=s;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
}break;
///////////////////////////////////////////
case 6:
{  int ii=0;int l;
int s=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{
 for(int k=i;k<i+3;k++)
```

```
for(int l=j;l<j+3;l++)
{if(ii==1||ii==3||ii==5||ii==7)
s=s+gr[k][l]*(0-mask[ii]);
else
s=s+gr[k][l]*mask[ii];
ii++;
}
a[i+1][j+1]=s;
s=0;
ii=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+100,i,a[i][j]);
}break;
//////////////////////////////////////////
case 7:
{  int l;
int ss=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{int x=0;
for(int k=i;k<i+2;k++)
for(int l=j;l<j+2;l++)
{z1[x]=gr[k][l];
x++;
}
int s1=fabs(z1[2]-z1[1]);
int s2=fabs(z1[3]-z1[0]);
ss=s1+s2;
a[i+1][j+1]=ss;
ss=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
}break;
////////////////////////////////////
case 8:
{  int z=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{
```

```
for(int k=i;k<i+3;k++)
for(int l=j;l<j+3;l++)
{s=s+(gr[k][l]*mask11[z]);
x=x+(gr[k][l]*mask12[z]);
z++;
}
int ss=sqrt((s*s)+(x*x));
a[i+1][j+1]=ss;
s=0;z=0;x=0;ss=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
}break;
//////////////////////////////
case 9:
{  int z=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{
for(int k=i;k<i+3;k++)
for(int l=j;l<j+3;l++)
{s=s+(gr[k][l]*n1[z]);
x=x+(gr[k][l]*n2[z]);
z++;
}
int ss=sqrt((s*s)+(x*x));
a[i+1][j+1]=ss;
s=0;z=0;x=0;ss=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
}break;
///////////////////////////////////
case 10:
{int  s0=0;int s1=0;int s2=0;
int s3=0;int s4=0;int s5=0;
int s6=0;int s7=0;
int z=0;
for(i=0;i<99;i++)
for(j=0;j<99;j++)
{z=0;
```

```
for(int k=i;k<i+3;k++)
for(int l=j;l<j+3;l++)
{
s0=s0+(gr[k][l]*l0[z]);
s1=s1+(gr[k][l]*l1[z]);
s2=s2+(gr[k][l]*l2[z]);
s3=s3+(gr[k][l]*l3[z]);
s4=s4+(gr[k][l]*l4[z]);
s5=s5+(gr[k][l]*l5[z]);
s6=s6+(gr[k][l]*l6[z]);
s7=s7+(gr[k][l]*l7[z]);
z++;
}
max[0]=s0;
max[1]=s1;
max[2]=s2;
max[3]=s3;
max[4]=s4;
max[5]=s5;
max[6]=s6;
max[7]=s7;
x=0;
for(int t=0;t<8;t++)
{if(max[t]>=x)
x=max[t];}
a[i+1][j+1]=x;
  z=0;
 s0=0; s1=0;
 s2=0; s3=0; s4=0; s5=0;
 s6=0; s7=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
}break;
//////////////////////////////
case 11:
{ int s0=0;
int s1=0;
int s2=0;
int s3=0;
int s4=0;
int s5=0;int s6=0;int s7=0;
```

```
 int z=0;
 for(i=0;i<99;i++)
 for(j=0;j<99;j++)
 { z=0;
 for(int k=i;k<i+3;k++)
 for(int l=j;l<j+3;l++)
 {
 s0=s0+(gr[k][l]*f0[z]);
 s1=s1+(gr[k][l]*f1[z]);
 s2=s2+(gr[k][l]*f2[z]);
 s3=s3+(gr[k][l]*f3[z]);
 s4=s4+(gr[k][l]*f4[z]);
 s5=s5+(gr[k][l]*f5[z]);
 s6=s6+(gr[k][l]*f6[z]);
 s7=s7+(gr[k][l]*f7[z]);
 z++;
 }
 max[0]=s0;
 max[1]=s1;
 max[2]=s2;
 max[3]=s3;
 max[4]=s4;
 max[5]=s5;
 max[6]=s6;
 max[7]=s7;
 x=0;
 for(int t=0;t<8;t++)
 {if(max[t]>=x)
 x=max[t];}
 a[i+1][j+1]=x;
   z=0;
  s0=0; s1=0;
  s2=0; s3=0; s4=0; s5=0;
  s6=0; s7=0;
 }
 for(i=0;i<100;i++)
 for(j=0;j<100;j++)
 putpixel(j+110,i,a[i][j]);
 }
/*////////////////////////////////////////////////*/
case 12:
{int n=0;
for(int k1=0;k1<256;k1++)
```

```
{ n=0;
for (i=0;i<bmp.depth;i++)
for(j=0;j<bmp.width;j++)
 if(gr[i][j]==k1)
    n=n+1;
 g2[k1]=n;
 }
 for(i=0;i<258;i++)
 {
   putpixel(258,i,111) ;
   putpixel(i,199,111);
   }
 for(k1=0;k1<255;k1++)
   for(int k=199;k>(199-g2[k1]);k--)
     putpixel(k1,k,k1+10);
   }break;
   /****************************/
   case 13:  /*prority for all image*/
   {
   closegraph();
   pro();
   }  break;
   /********************************/
   case 14:
  { closegraph();
  int s=0;
  int m=bmp.depth*bmp.width;
  for (i=0;i<bmp.depth;i++)
  for(j=0;j<bmp.width;j++)
    s=s+gr[i][j];
float  mean=s/m;
   cout<<"mean:"<<mean;
  }} /*break;
  /**********************************/

  closegraph();
}
    /*******************************************/
 void pro()
 {float p[10000];
 float c=0;
 int m=bmp.depth*bmp.width;
 for(int k1=0;k1<256;k1++)
```

```
 {
  for (i=0;i<bmp.depth;i++)
  for(j=0;j<bmp.width;j++)
   if(gr[i][j]==k1)
     n=n+1;
 p[k1]=n/m;
 c=c+p[k1];
 cout<<"p["<<k1<<"]:="<<p[k1]<<" ";

 }
  cout <<"\n"<<c;
}
```

برنامج لإيجاد الهستوغرام

```
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<float.h>
#include<iostream.h>
#include<math.h>
#include<stdio.h>
// process(int fr,int er,int fc,int ec);
typedef struct{
char id[2];
long filesize;
int reserved[2];
long headersize;
long infoSize;
long width;
long depth;
int biPlanes;
int bits;
long bicompression;
long biSizeImage;
long biXPelsPerMeter;
long biYPelsPerMeter;
long biClrUsed;
```

```
long biClrImportant;
}BMPHEAD;
/************************************************************
*****************/
FILE *fp,*fn,*fd;
/************************************************************
*****************/
int q,j,i=0,m,n,s,x,l,kl;
int gdriver,gmode;
unsigned char far
gr[100][100],gr1[100][100],gr2[100][100],gr3[200][200],c,kc;
unsigned char color[256][3];
BMPHEAD bmp;
int b[4];
float avr;
float w;
float s1;
int thre=3;
//int b[8];
int b1[8];
int hist[255];
static int r[9]={1,2,4,8,16,32,64};
static float mask[9]={0,1,0,0,1,0,0,-1,0};
static float mask24[9]={0.25,0.5,0.25,0.5,1,0.5,0.25,0.5,0.25};
static float mask11[4]={1,1,1,1};
static float mask1[9]={0,-1,0,-1,0,-1,0};
//////////////////////////////////////////////////////////////////////////////////////
int huge detectsvga(void)
{
return 0;
}
int matherr(struct exception *a)
{
if(a->type==DOMAIN)
if(!strcmp(a->name,"sqrt"))
a->retval=sqrt(-(a->arg1));
return 1;
}
/* return 0;
}
int matherr1 (struct exception *a)
{
if(a->type==DOMAIN)
```

```
if(!strcmp(a->name,"sqrt"))
a->retval=sqrt(-(a->argl));
return 1;
} */
void init_graph(void)
{
int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=DETECT;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
/////////////repeat/////////////////////////
void repeet(unsigned char gr[100][100])
{
int n=100,m=100,k1;
init_graph();
k1=0;
for(i=0;i<n*2;i=i+2)
{
for(j=0;j<m;j++)
{
gr3[i][j]=gr[k1][j];
putpixel(j,i,gr3[i][j]);
gr3[i+1][j]=gr[k1][j];
putpixel(j,i+1,gr3[i][j]);
}k1=k1+1;
}getchar();
getchar();
}
//////////////////////////////////////////////////////////////////////////
void display(){
init_graph();
setcolor(100);
line(0,199,256,199);
line(0,199,256,199);
line(0,90,0,199);
line(256,90,256,199);
int max=hist[0];
for(i=1;i<255;i++)if(hist[i]>max)
max=hist[i];
```

```
int d;
d=max/100;
if(d<1)
d=1;
cout<<"scale of(y/d)axis="<<d;
setcolor(254);
for(i=0;i<255;i++)
line(i+1,199,i+1,199-(hist[i]/d));
getchar();
getchar();
}
void histogram()
{
init_graph();
for(i=0;i<255;i++)hist[i]=0;
for(i=0;i<bmp.depth;i++)
for(j=0;j<bmp.width;j++)
{
int x;
x=gr[i][j];
hist[x]++;
}
display();
}
/////////////////////array/of /image/////////////////////////////////////////////////////////
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{
fseek(fp,lw,0);
fread(bmp_line,nw,l,fp);
for(j=0;j<=bmp.width-1;j++)
{
gr[i][j]=bmp_line[j];
putpixel(j+120,i+30,bmp_line[j]);
}lw=lw-nw;
}
//////////////////////////////////////////////////////////////////////////////
fclose(fp);
getchar();
getchar();
closegraph();
```

```
getchar();
for(i=0;i<bmp.depth;i++)
for(j=0;j<bmp.width,j++)
putpixel(j,i,255-gr[i][j]);
getchar();
//////////////////////////////////////////////////////////////////////////
// histogram
read(gr);
histogram();
closegraph();
//mean
s1=0.0;
//for(i=0;i<bmp.depth;i++)
float n=bmp.depth*bmp.width;
for(i=0;i<255;i++)
s1+=(hist[i]/(n)*i);
float mean=s1;
// float mean=(1/bmp.depth+bmp.width)*s;
cout<<"mean="<<endl;
//stander divation
s1=0.0;
for(int g=0;g<=255;g++)
s1+=(((g-mean)*(g-mean))*hist[g]/(n)));
float sd=sqrt(s1);
/*  for(i=0;i<bmp.depth;i++);
  for(j=0;j<bmp.width;j++)
  if(gr[i][j]==g)
  no++;
  float p=no/bmp.depth+bmp.width;
  s*=p;
  }*/
  cout<<endl<<"sd="<<(sd);
  }
```

برنامج معالجة المناطق الداخلية:

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
```

```
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/*********************************/
FILE  *fp,*fn,*fd,*ff;
/********************************/
int compute(int);
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],gr1[100][100];
unsigned char far a[100][100];
unsigned char far a1[100][200],a2[200][200];
unsigned char far n1[100][100];
unsigned char color[256][3];
  unsigned char far b1[200][200];
     unsigned char far b2[205][205];
    unsigned char far b3[200][200];
float mask1[9]={0.25,0.5,0.25,0.5,1,0.5,0.25,0.5,0.25};
bmphead bmp;
float mask[3][3];
/*int k=0;
for(int ii=0;ii<3;ii++)
```

```
{for(int jj=0;jj<3;jj++)
mask[ii][jj]=mask1[k];k++;}*/
///////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
int compute(int x1)
{int x[9];
for(i=0;i<8;i++)
{x[i]=x1%2;
x1=x1/2;
}
for(i=0;i<8;i++)
{if(x[i]==0)
   x[i]=1 ;
if (x[i]==1)
  x[i]=0;
 }
 int s=0;
 int j=7;
for(i=0;i<8;i++)
{int l=pow(2,j);
s=s+(l*x[i]);
j--;
}
return s;
}
///////////////////////////////
void main()
{clrscr();
int no;
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25],file_name1[25];
long lw,nw;
```

```
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
     printf("enter file name to be loaded >>   ");
scanf("%s",file_name1);
     cout<<"##########"<<"\n";
     cout<<"**********"<<"\n";
     cout<<"1-cut[1]"<<"\n";
     cout<<"2-cut[2]"<<"\n";
     cout<<"3-cut[3]"<<"\n";
     cout<<"4-cut[4]"<<"\n";
     cout<<"5-zoero order zooming"<<"\n";
     cout<<"6-average zooming"<<"\n";
     cout<<"7-first convolution"<<"\n";
     cout<<"8-general zomming"<<"\n";
     cout<<"9-add"<<"\n";
     cout<<"10-sub"<<"\n";
     cout<<"11-mult"<<"\n";
     cout<<"12-div"<<"\n";
     cout<<"13-and"<<"\n";
     cout<<"14-or"<<"\n";
     cout<<"15-not"<<"\n";
     cout<<"**********************"<<"\n";
     cout<<"######################"<<"\n";
     cout<<"****inter yout selection****  ";
     cin>>no;
////////////array/of/image////////////////
```

```
void d();
{int r=99;
/*unsigned char far*/
cout<<r;
}
long int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
getchar();
switch(no)
{ case 1:
  for(i=0;i<50;i++)
  for(j=0;j<50;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   ///////////////////////////////
   case 2:
  for(i=0;i<50;i++)
  for(j=50;j<=100;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   ///////////////////////////
   case 3:
  for(i=50;i<=100;i++)
  for(j=0;j<50;j++)
  {gr1[i][j]=gr[i][j];
   putpixel(j+110,i+100,gr1[i][j]);
   }break;
   //////////////////////////
   case 4:
  for(i=50;i<=100;i++)
```

```
for(j=50;j<=100;j++)
{gr1[i][j]=gr[i][j];
 putpixel(j+110,i+100,gr1[i][j]);
 }break;
 ////////////////////////
 case 5:
 {for(i=0;i<=bmp.depth;i++)
  {x1=0;
  for(j=0;j<=bmp.width;j++)
  {a1[i][x1]=gr[i][j];
   x1++;
   a1[i][x1]=gr[i][j];
   x1++;
   }}
   for(i=0;i<=2*bmp.depth;i++)
   {y1=0;
   for(j=0;j<=bmp.width;j++)
   {a2[y1][i]=a1[j][i];
   y1++;
   a2[y1][i]=a1[j][i];
   y1++;}}
   for(i=0;i<=2*bmp.depth;i++)
   for(j=0;j<=2*bmp.width;j++)
   putpixel(j+101,i,a2[i][j]);
   }break;
   ////////////////////////////
 case 6:
 {for(i=0;i<=bmp.depth;i++)
  {x1=0;
  for(j=0;j<=bmp.width;j++)
  {a1[i][x1]=gr[i][j];
   x1++;
   a1[i][x1]=(gr[i][j]+gr[i][j+1])/2;
   x1++;
   }}
   for(i=0;i<=2*bmp.depth;i++)
   {y1=0;
   for(j=0;j<=bmp.width;j++)
   {a2[y1][i]=a1[j][i];
   y1++;
   a2[y1][i]=(a1[j][i]+a1[j+1][i])/2;
   y1++;}}
   for(i=0;i<=2*bmp.depth;i++)
```

```
for(j=0;j<=2*bmp.width;j++)
putpixel(j+101,i,a2[i][j]);
}break;
//////////////////////
case 7:
{int ii=0;

for(i=0;i<100;i++)
{for(j=0;j<100;j++)
b1[ii][j]=0;
ii++;
for(j=0;j<100;j++)
b1[ii][j]=gr[i][j];
ii++;
}
for(j=0;j<100;j++)
b1[ii][j];
int jj=0;
for(j=0;j<100;j++)
{for(i=0;i<201;i++)
b2[i][jj]=0;
jj++;
for(j=0;j<201;j++)
b2[i][jj]=b1[i][j];
jj++;
}
ii=0;
int s=0;int k,l;
for(i=0;i<199;i++)
for(j=0;j<199;j++)
{for(k=i;k<i+3;k++)
for(l=j;l<j+3;l++)
{s=s+(b2[k][l]*mask1[ii]);
i++;
}
b3[i][j]=s;
ii=0; s=0;
}
for(i=0;i<199;i++)
for(j=0;j<199;j++)
putpixel(j+150,i,b3[i][j]);
    }break;
    /////////////////////
```

```
          case 15:
          {for(i=0;i<100;i++)
           for(j=0;j<100;j++)
           {int l=compute(gr[i][j]);
            n1[i][j]=l;
            putpixel(j+100,i,n1[i][j]);
            }
            }}
          fclose(fp);
  ff=fopen(file_name1,"rb");
fread(&bmp,1,sizeof(bmphead),ff);

  for(i=0;i< 255;++i)
    {c1=fgetc(ff);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(ff);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(ff);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(ff);
     }
     nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(ff,lw,0);
  fread(bmp_line,nw,1,ff);
  for(j=0;j<=bmp.width-1;j++)
  {a[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  //histogram();
  getchar();
  getchar();
  closegraph();
  }
  //return(0);
  /////////////////////////////
```

برنامج لاستخدام mean filter :

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/*********************************/
FILE  *fp,*fn,*fd;
/********************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
float mask[9]={0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};
unsigned char color[256][3];
bmphead bmp;
///////////////////////////////
int detectsvga(void)
```

```
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
```

```
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  int ii=0;int l;
  int s=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {
  for(int k=i;k<i+3;k++)
  for(int l=j;l<j+3;l++)
  {
  s=s+gr[k][l]*mask[ii];
  ii++;
  }
  a[i+1][j+1]=s;
  s=0;
  ii=0;
  }
  for(i=0;i<100;i++)
  for(j=0;j<100;j++)
  putpixel(j+100,i,a[i][j]);
  fclose(fp);
  //histogram();
  getchar();
  getchar();
  getchar();
  closegraph();
  }
  //return(0);
  //////////////////////////////
```

برنامج مرشح prewit filter :

```
#include<math.h>
#include<iostream.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
```

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,i=0,j=0,m,n,x1,y1;
int gdriver,gmode;
int s;
unsigned char far gr[100][100],a[100][100];
 int mask1[9]={(-1,-1,-1),(0,0,0),(1,1,1)};
 int  mask2[9]={(-1,-1,-1),(0,0,0),(1,1,1)};
unsigned char color[256][3];
bmphead bmp;
//////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
```

```
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
//////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
```

```
 lw=lw-nw;
}
 int z=0;
 for(i=0;i<99;i++)
 for(j=0;j<99;j++)
 {
 for(int k=i;k<i+3;k++)
 for(int l=j;l<j+3;l++)
 {s=s+(gr[k][l]*mask1[z]);
 x=x+(gr[k][l]*mask2[z]);
 z++;
 }
 int ss=sqrt((s*s)+(x*x));
 a[i+1][j+1]=ss;
 s=0;z=0;x=0;ss=0;
 }
 for(i=0;i<100;i++)
 for(j=0;j<100;j++)
 putpixel(j+110,i,a[i][j]);
 fclose(fp);
 //histogram();
 getchar();
 getchar();
 getchar();
 closegraph();
 }
 //return(0);
 /////////////////////////////
```

برنامج مرشحrobert1 :

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
```

```
#include<io.h>
#include<float.h>
#include<math.h>
typedef struct{
char id[2];
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
unsigned char far z[4];
unsigned char color[256][3];
bmphead bmp;
////////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
```

```
///////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
/////////////array/of/image////////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  int l;
  int ss=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
```

```
{int x=0;
for(int k=i;k<i+2;k++)
for(int l=j;l<j+2;l++)
{z[x]=gr[k][l];
x++;
}
int s1=fabs(z[2]-z[1]);
int s2=fabs(z[3]-z[0]);
ss=s1+s2;
a[i+1][j+1]=ss;
ss=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+110,i,a[i][j]);
fclose(fp);
//histogram();
getchar();
getchar();
getchar();
closegraph();
}
//return(0);
//////////////////////////////
```

برنامج مرشحRobert :

```
#include<iostream.h>
#include<math.h>
#include<alloc.h>
#include<graphics.h>
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#include<io.h>
#include<float.h>
#include<math.h>
typedef struct{
char id[2];
```

```
long filesize;
int reseued[2];
long headersize;
long infosize;
long width;
long depth;
int biplanes;
int bits;
long bicompression;
long bisizeimage;
long bixpelspermeter;
long biypelspermeter;
long biclrused;
long biclrimportant;
}
bmphead;
/********************************/
FILE  *fp,*fn,*fd;
/*******************************/
int q,i=0,j=0,m,n,s,x1,y1;
int gdriver,gmode;
unsigned char far gr[100][100],a[100][100];
unsigned char far z[4];
unsigned char color[256][3];
bmphead bmp;
////////////////////////////////
int detectsvga(void)
{return 0;
}
void init_graph(void)
{int i;
gdriver=installuserdriver("svga256",detectsvga);
//gdriver=detect;
//gmode=4;
initgraph(&gdriver,&gmode,"");
for(i=0;i<255;i++)
setrgbpalette(i,color[i][2],color[i][1],color[i][0]);
}
///////////////////////////////
void main()
{clrscr();
char c1,c2,c3,c4;
char bmp_line[1024],file_name[25];
```

```
long lw,nw;
float n,x;
clrscr();
printf("enter file name to be loaded >>   ");
scanf("%s",file_name);
fp=fopen(file_name,"rb");
fread(&bmp,1,sizeof(bmphead),fp);

  for(i=0;i< 255;++i)
    {c1=fgetc(fp);
     c1=c1>>2;
     color[i][0]=c1;
     c2=fgetc(fp);
     c2=c2>>2;
     color[i][1]=c2;
     c3=fgetc(fp);
     c3=c3>>2;
     color[i][2]=c3;
     fgetc(fp);
     }
////////////array/of/image//////////////////
int x1,y1;
init_graph();
nw=4*((bmp.width+3)/4);
lw=(bmp.depth-1)*nw+1078;
for(i=0;i<=bmp.depth-1;i++)
{ fseek(fp,lw,0);
  fread(bmp_line,nw,1,fp);
  for(j=0;j<=bmp.width-1;j++)
  {gr[i][j]=bmp_line[j];
  putpixel(j,i,bmp_line[j]);
  }
  lw=lw-nw;
}
  int l;
  int ss=0;
  for(i=0;i<99;i++)
  for(j=0;j<99;j++)
  {int x=0;
  for(int k=i;k<i+2;k++)
  for(int l=j;l<j+2;l++)
  {z[x]=gr[k][l];
  x++;
```

```
}
int s1=fabs(z[2]-z[1]);
int s2=fab(z[3]-z[0]);
ss=s1+s2;
a[i+1][j+n]=ss;
ss=0;
}
for(i=0;i<100;i++)
for(j=0;j<100;j++)
putpixel(j+100,i,a[i][j]);
fclose(fp);
//histogram();
getchar();
getchar();
getchar();
closegraph();
}
//return(0);
//////////////////////////////
```

References:

on, D. J.Schneberk, M. F. Skeate (1990). Geometric Effects in Tomographic Reconstruction. Lawrence Livermore National Laboratory Rep. UCRL-ID-105130.

M. F. Barnsley, L. P. Hurd (1993). Fractal Image Compression. Peters, Wellesley, MA.

G. Bertrand, J-C. Everat, M. Couprie (1997). Image segmentation through operators based on topology. J. Electron. Imaging 6(4):395-405.

M. A. Bassiouni, N. Tzannes, M. Tzannes (1993). High-fidelity integrated lossless/lossy compression and reconstruction of images. Opt. Eng. 32(8):1848-1853.

F. R. Boddeke, L. J. van Vliet, H. Netten, I. T. Young (1994). Autofocusing in microscopy based on the OTF and sampling. Bioimaging 2:193-203.

D. S. Bright, D. E. Newbury, E. B. Steel (1998). Visibility of objects in computer simulations of noisy micrographs. J. Microsc. 189(1):25-42.

W. A. Carrington (1990). Image restoration in 3D microscopy with limited data, in Bioimaging and Two Dimensional Spectroscopy, Proc. SPIE, Vol. 1205 (L. C. Smith, Ed.), 72-83.
.

P. Chieco, A. Jonker, C. Melchiorri, G. Vanni, C. J. F. van Noorden (1994). A user’s guide for avoiding errors in absorbance image cytometry. Histochem. J. 26:1-19.

C. K. Chui (1992). An Introduction to Wavelets. Academic Press, London.

J. Cookson (1994). Three-Dimensional Reconstruction in Microscopy. Proc. R. Microsc. Soc. 29(1) Jan., 1994, pp. 3-10.

D. M. Coppola, H. R. Purves, A. N. McCoy, D. Purves (1998). The distribution of oriented contours in the real world. Proc. Natl. Acad. Sci. 95:4002-4006.

D. G. Daut, D. Zhao, J. Wu (1993). Double predictor differential pulse coded modulation algorithm for image data compression. Opt. Eng. 32(7):1514-1523.

J. Davidson (1991). Thinning and skeletonization: a tutorial and overview, in Digital Image Processing: Fundamentals and Applications (E. Dougherty, Ed.). Marcel Dekker, New York.

D. DeMandolx, J. Davoust (1997). Multicolor analysis and local image correlation in confocal microscopy. J. Microsc. 185: 21-36.

G. Diaz, D. Quacci, C. Dell'Orbo (1990). Recognition of cell surface modulation by elliptic Fourier analysis. Comput. Methods Programs Biomed. 31: 57-62.

M. Dietzsch, K. Papenfuss, T. Hartmann (1997). The MOTIF method (ISO 12085) — a suitable description for functional manufactural and metrological requirements, in 7th Int. Conf. Metrol. Prop. Eng. Surf. (B. G. Rosen, R. J. Crafoord, Eds.), Chalmers Univ., Göteborg, Sweden, pp. 231-238.

E. R. Dougherty, J. Astola (1994). An Introduction to Nonlinear Image Processing. SPIE, Bellingham, WA.

M. W. Mitchell, D. A. Bonnell (1990). Quantitative topographic analysis of fractal surfaces by scanning tunneling microscopy. J. Mat. Res. 5(10):2244-2254.

J. R. Monck, A. F. Oberhauser, T. J. Keating, J. M. Hernandez (1992). Thin-section ratiometric Ca2+ images obtained by optical sectioning of Fura-2 loaded mast cells. J. Cell Biol. 116:745-759.

R. B. Mott (1995). Position-tagged spectrometry, a new approach for EDS spectrum imaging. Proc. Microsc. Microanal., p. 595.

Jones and Begall, NY.

K. S. Nathan, J. C. Curlander (1990). Reducing speckle in onelook SAR images. NASA Tech. Briefs, Feb:70.

W. Niblack (Ed.) (1993). Storage and retrieval for image and video databases. SPIE Proc., Vol. 1908.

A. Nicoulin, M. Mattavelli, W. Li, M. Kunt (1993). Subband image coding using jointly localized filter banks and entropy coding based on vector quantization. Opt. Eng. 32(7):1430-1450.

K. Oistämö, Y. Neuvo (1990). Vector median operations for color image processing. Nonlinear Image Processing (E. J. Delp, Ed.). SPIE Proc. 1247:2-12
.
C. K. Olsson (1993). Image Processing Methods in Materials Science. Ph. D. Thesis, Technical University of Denmark, Lyngby, Denmark.

W. K. Pratt (1991). Digital Image Processing, Second Ed. Wiley, New York.

T. Prettyman, R. Gardner, J. Russ, K. Verghese (1991). On the performance of a combined transmission and scattering approach to industrial computed tomography. Advances in X-Ray Analysis, Vol. 35. Plenum Press, New York.

C. F. Quate (1994). The AFM as a tool for surface imaging. Surf. Sci. (Netherlands) 299-300, 980-95.

M. G. Reed, C. V. Howard, C. G. Shelton (1997). Confocal imaging and second-order stereological analysis of a liquid foam. J. Microsc. 185(3):313-320.

M. G. Reed, C. V. Howard (1997). Edge corrected estimates of the nearest neighbor function for three-dimensional point patterns. J. Microsc. (in press).

M. G. Reed, C. V. Howard (1998). Unbiased Stereology. Bios Scientific Pub., Oxford.

A. A. Reeves, Optimized Fast Hartley Transform with Applications in Image Processing, Thesis, Dartmouth University, March 1990.

G. X. Ritter, J. N. Wilson (1996). Handbook of Computer Vision Algorithms in Image Algebra. CRC Press, Boca Raton, FL.

B. G. Rosen, R. J. Crafoord, (Eds.) (1997). Metrology and Properties of Engineering Surfaces. Chalmers University, Göteborg Sweden.

J. C. Russ (1991). Multiband thresholding of images. J. Comput. Assist. Microsc. 3(2):77-96.

J. C. Russ (1993). JPEG Image Compression and Image Analysis. J. Comput. Assist. Microsc. 5(3):237-244.

J. C. Russ (1993). Method and application for ANDing features in binary images. J. Comput. Assist. Microsc. 5(4):265-272.

J. C. Russ (1995). Thresholding images. J. Comput. Assist. Microsc. 7(3):41-164.

J. C. Russ (1995). Designing kernels for image filtering. J. Comput. Assist. Microsc. 7(4):179-190.

J. C. Russ (1995). Optimal greyscale images. J. Comput. Assist. Microsc. 7(4):221-234.

J. C. Russ (1995f). Segmenting touching hollow features. J. Comput. Assist. Microsc. 7(4):253-261.

J. C. Russ (1997). Fractal dimension measurement of engineering surfaces, in 7th Int. Conf. Metrol. Prop. Eng. Surf. (B. G. Rosen, R. J. Crafoord, Eds.), Chalmers University, Göteborg, Sweden, 170-174.

B. D. Smith (1990). Cone-beam tomography: recent advances and a tutorial review. SPIE Opt. Eng. 29:5.

D. L. Snyder, T. J. Schutz, J. A. O'Sullivan (1992). Deblurring subject to nonnegative constraints. IEEE Trans. Signal Process. 40:1143-1150.

S. Srinivasan, J. C. Russ, R. O. Scattergood (1990). Fractal analysis of erosion surfaces. J. Mat. Res. 5(11):2616-2619.

J. A. Stark, W. J. Fitzgerald (1996). An alternative algorithm for adaptive histogram equalization. Comput. Vis. Graph. Image Process. 56(2):180-185.

J. A. Storer (1992). Image and Text Compression. Kluwer Academic Publishers, New York.

R. E. Swing (1997). An Introduction to Microdensitometry. SPIE Press, Bellingham, WA.

J. G. Verly, R. L. Delanoy (1993). Some principles and applications of adaptive mathematical morphology for range imagery. Opt. Eng. 32(12):3295-3306.

H. Verschueren, B. Houben, J. De Braekeleer, J. De Wit, D. Roggen, P. De Baetselier (1993). Methods for computer assisted analysis of lymphoid cell shape and motility, including Fourier analysis of cell outlines. J. Immunol. Methods 163: 99-113.

J. S. Villarrubia (1994). Morphological estimation of tip geometry for scanned probe microscopy. Surf. Sci. 321:287-300.

J. S. Villarrubia (1996). Scanned probe microscope tip characterization without calibrated tip characterizers. J. Vac. Sci. Technol. B14:1518-1521.

G. Wang, T. H. Lin, P. C. Cheng, D. M. Shinozaki, H. Kim (1991). Scanning cone-beam reconstruction algorithms for X-ray microtomography. SPIE Scanning Microsc. Instrument. 1556:99.

A. Wen, C. Lu (1993). Hybrid vector quantization. Opt. Eng. 32(7):1496-1502.

D. J. Whitehouse (1994). Precision — The Handbook of Surface Metrology. Institute of Physics Publishing, Bristol.

H. K. Wickramasinghe (1991). Scanned probes old and new. AIP Conf. Proc. (USA), 9-22.

Z. Wang (1990). Principles of Photogrammetry (with Remote Sensing). Press of Wuhan Technical University of Surveying and Mapping, Beijing.

G. Wolf (1991). Usage of global information and a priori knowledge for object isolation. Proc. 8th Int. Congr. Stereol., Irvine, CA, 56.

S. H. Wong, S. F. Yau (1998). Linear neural network for the solution of limited angle problems in computer-aided tomography. J. Electron. Imaging 7(1):70-78.

B. P. Wrobel (1991). Least-squares methods for surface reconstruction from images. ISPRS J. Photogramm. Remote Sensing, 46:67-84.

S. Wu, A. Gersho (1993). Lapped vector quantization of images. Opt. Eng. 32(7):1489-1495.

R. W. Young, N. G. Kingsbury (1993). Video compression using lapped transforms for motion estimation/compensation and coding. Opt. Eng. 32(7):1451-1463.

X. Zhou, E. Dorrer (1994). An automatic image matching algorithm based on wavelet decomposition, ISPRS Int. Arch. Photogramm. Remote Sensing 30(3/2):951-960.

رقم الإيداع في دار الكتب والوثائق الوطنية ببغداد (١٠٥٢) لسنة ٢٠٠٨